L Number	Hits	Search Text	DB	Time stamp
1	18713	transmit$ with (backup or archiv$ or recover$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 07:42
2	36380	transmit$ same(backup or archiv$ or recover$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 07:43
3	992	transmit$ adj(backup or archiv$ or recover$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 07:43
4	18	(transmit$ adj(backup or archiv$ or recover$)) and database and (stream$ or array) and (update with transaction)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 08:12
5	12620	"707" and "709" and "705"	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 08:00
6	157	("707" and "709" and "705") and (backup or archiv$ or recover$) and logs and transaction and (parallel or simultanously)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 08:01
7	1	(("707" and "709" and "705") and (backup or archiv$ or recover$) and logs and transaction and (parallel or simultanously)) and (transmit$ adj(backup or archiv$ or recover$))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 08:01
8	2	("5403639").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 09:35
9	1	("RE37857").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 10:32
10	26338	compar$ with (backup or recover$ pr archiv$)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 10:33
11	31868	(corrupt$ or miss$) with (file or data)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 10:34
12	872	(compar$ with (backup or recover$ pr archiv$)) and ((corrupt$ or miss$) with (file or data))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:47
14	104	copy$ and transmit$ and (copy$ with corrupt$) and check$3 and logs	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:45
15	5	((compar$ with (backup or recover$ pr archiv$)) and ((corrupt$ or miss$) with (file or data))) and (copy$ and transmit$ and (copy$ with corrupt$) and check$3 and logs)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 10:39

16	358	copy$ and transmit$ and (copy$ with corrupt$) and check$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 10:38
17	13	((compar$ with (backup or recover$ pr archiv$)) and ((corrupt$ or miss$) with (file or data))) and (copy$ and transmit$ and (copy$ with corrupt$) and check$3)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:46
18	2	("5812398").PN.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 10:58
19	0	(copy$ and transmit$ and (copy$ with corrupt$) and check$3 and logs) and backup and rsync	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:45
20	35	(copy$ and transmit$ and (copy$ with corrupt$) and check$3 and logs) and backup and checksum	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:45
21	3	((compar$ with (backup or recover$ pr archiv$)) and ((corrupt$ or miss$) with (file or data))) and (copy$ and transmit$ and (copy$ with corrupt$) and check$3) and checksum	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:47
22	128	(compar$ with (backup or recover$ pr archiv$)) and ((corrupt$ or miss$) with (file or data)) and checksum	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/15 11:47

Search Results

Nothing Found

Your search for **[transmit <near/4> logs<AND>((database and compare and (corrupt or missing or lost)<AND>(((backup or recover or archive) and logs))))]** did not return any results.

You may revise it and try your search again below or click advanced search for more options.

```
transmit <near/4>
logs<AND>((database and compare and
(corrupt or missing or
lost)<AND>(((backup or recover or
archive) and logs ) )) )
```

SEARCH [Advanced Search] [Search Help/Tips]

Complete Search Help and Tips

The following characters have specialized meaning:

Special Characters	Description
, ( ) [	These characters end a text token.
= > < !	These characters end a text token because they signify the start of a field operator. (! is special: != ends a token.)
` @ \Q < { [!	These characters signify the start of a delimited token. These are terminated by the end character associated with the start character.

Your search matched **[0]** of **[950522]** documents.

You may refine your search by editing the current search expression or entering
a new one the text box. Then click search Again.

database and compare and (corrupt or missing or lost) and (backup or recover or archive)

Search Again

OR

Use your browser's back button to return to your original search page.

Results:

No documents matched your query.

Your search matched **3** of **950522** documents.

A maximum of **3** results are displayed, **15** to a page, sorted by **Relevance** in **descending** order.
You may refine your search by editing the current search expression or entering a new one the text box.

Then click **Search Again**.

database and compare and (corrupt or missing or lost) and (backup or recover or archive)

Search Again

Results:
Journal or Magazine = **JNL** Conference = **CNF** Standard = **STD**

1 **Intercomparison of drogued and undrogued drift buoys**
Pazan, S.E.;
OCEANS '96. MTS/IEEE. 'Prospects for the 21st Century'. Conference Proceedings , Volume: 2 , 23-26 Sept. 1996
Page(s): 864 -872 vol.2

[Abstract] [PDF Full-Text (832 KB)] **IEEE CNF**

2 **Robust word recognition for museum archive card indexing**
Lucas, S.M.; Tams, A.C.; Cho, S.J.; Ryu, S.; Downton, A.C.;
Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on , 10-13 Sept. 2001
Page(s): 144 -148

[Abstract] [PDF Full-Text (424 KB)] **IEEE CNF**

3 **Restoration scheme of mobility databases by mobility learning and prediction in PCS networks**
Joon-Min Gil; Chan Yeol Park; Chong-Sun Hwang; Doo-Soon Park; Jin Gon Shon; Young-Sik Jeong;
Selected Areas in Communications, IEEE Journal on , Volume: 19 Issue: 10 , Oct. 2001
Page(s): 1962 -1973

[Abstract] [PDF Full-Text (208 KB)] **IEEE JNL**

US006347322B1

(12) **United States Patent**
Bogantz et al.

(10) **Patent No.: US 6,347,322 B1**
(45) **Date of Patent: Feb. 12, 2002**

(54) **TRANSACTION STATE DATA REPLICATION BY TRANSACTION FORWARDING IN REPLICATED DATABASE SYSTEMS**

(75) Inventors: **Robert L. Bogantz**, Columbus; **Gary Michael Green**, Lancaster; **Sidney Dean Hester**, Pickerington, all of OH (US)

(73) Assignee: **Lucent Technologies Inc.**, Murray Hill, NJ (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/188,465**

(22) Filed: **Nov. 9, 1998**

(51) **Int. Cl.**[7] **G06F 12/00**
(52) **U.S. Cl.** **707/202**; 707/201; 707/10
(58) **Field of Search** 707/10, 202, 204, 707/201

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,819,159	A *	4/1989	Shipley et al.	714/19
5,036,518	A *	7/1991	Tseung	714/748
5,170,480	A *	12/1992	Mohan et al.	707/201
5,261,069	A *	11/1993	Wilkinson et al.	711/145
5,633,999	A *	5/1997	Clowes et al.	714/6
5,781,910	A *	7/1998	Gostanian et al.	707/201
5,781,912	A *	7/1998	Demers et al.	707/202
5,796,999	A *	8/1998	Azagury et al.	707/10
5,805,798	A *	9/1998	Kearns et al.	714/48
5,999,931	A *	12/1999	Breitbart et al.	707/10
6,122,630	A *	9/2000	Strickler et al.	707/8

OTHER PUBLICATIONS

King, R.P. et al. "Management of a Remote Backup Copy for Disaster Recovery", ACM Transactions on Database Systems, vol. 16, No. 2, Jun. 1991, pp. 338–368.*

King et al., "Management of a Remote Backup Copy for Diaster Recovery", ACM Transactions on Database Systems, vol. 16, No. 2, Jun. 1991, pp. 338–368.*

Singhal, Mukesh, "Update Transport: A New Technique for Update Synchronization in Replicated Database Systems", IEEE Transactions on Software Engineering, vol. 16, No. 12, Dec. 1990, pp. 1325–1336.*

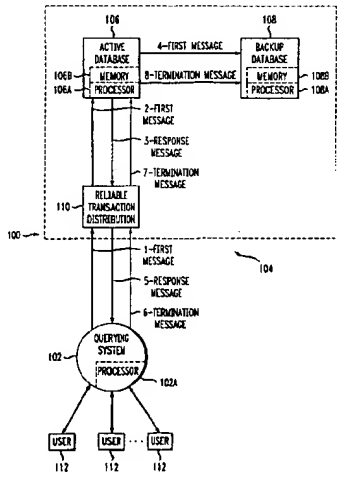
(List continued on next page.)

Primary Examiner—Jean R. Homere
Assistant Examiner—Luke S Wassum
(74) *Attorney, Agent, or Firm*—Stevens Showalter LLP

(57) **ABSTRACT**

A transaction processing system comprises a querying system and a logical database having an active database and a backup database. The querying system transmits a message for a transaction to the logical database for processing. The message is transmitted to the active database where the message is processed. The active database creates transaction state data based, in part, on the message. The active database transmits a response message to the querying system and forwards the original message to the backup database. The backup database processes the original message and creates its own transaction state data. The transaction state data in the backup database operationally matches the transaction state data in the active database so that if the active database fails, the backup database includes the requisite transaction state data necessary to complete the transaction. The querying system processes the response message and transmits a termination message to the logical database for processing. The termination message is transmitted to the active database and processed. The termination message is forwarded to the backup database from the active database and similarly processed. The transaction is then complete and both databases are fully replicated.

36 Claims, 4 Drawing Sheets



OTHER PUBLICATIONS

Schneider, Fred B. "Implementing Fault–Tolerant Services Using the State Machine Approach: A Tutorial", ACM Computing Surveys, vol. 22, No. 4, Dec. 1990, pp. 299–319.*

King et al. "Management of a Remote Backup Copy for Disaster Recovery", ACM Transactions on Database Systems, vol. 16, No. 2, Jun. 1991, pp. 338–368.*

Singhal, Mukesh. "Update Transport: A New Technique for Update Synchronization in Replicated Database Systems", IEEE Transactions on Software Engineering, vol. 16, No. 12, Dec. 1990, pp. 1325–1336.*

Rusinkiewicz, M., and D. Georgakopoulos. "Transaction Management in a Distributed Database System for Local Area Networks", Proceedings of the 7th Symposium on Reliable Distributed Systems, Oct. 10–12, 1988. pp. 177–182.*

Singhal, Mukesh. "A Fully–Distributed Approach to Concurrancy Control in Replicated Database Systems", Proceedings of the 12th International Conference on Computer Software and Applications, Oct. 5–7, 1988. pp. 353–360.*

Rusinkiewciz, M., and D. Georgakopoulos. "Request II: A Distributed Database System for Local Area Networks", Proceedings of the 1986 IEEE Fall Joint Computer Conference, pp. 1179–1188.*

Bernstein, P., and N. Goodman. "An Algorithm for Concurrancy Control and Recovery in Replicated Distributed Databases", ACM Transactions on Database Systems, vol. 9, No. 4, Dec. 1984, pp. 596–615.*

* cited by examiner

FIG. 1

BACKGROUND ART

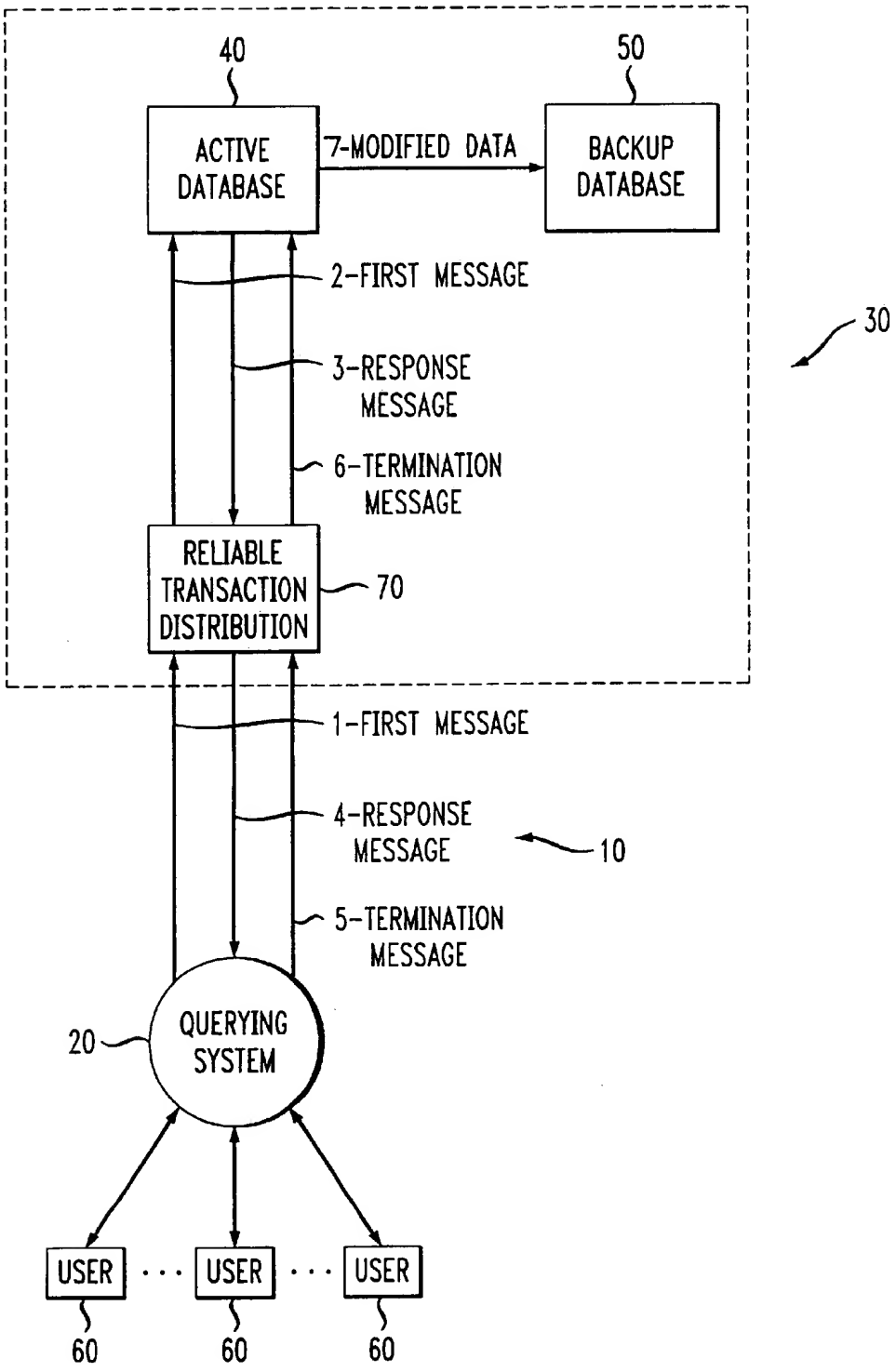


FIG. 2

BACKGROUND ART

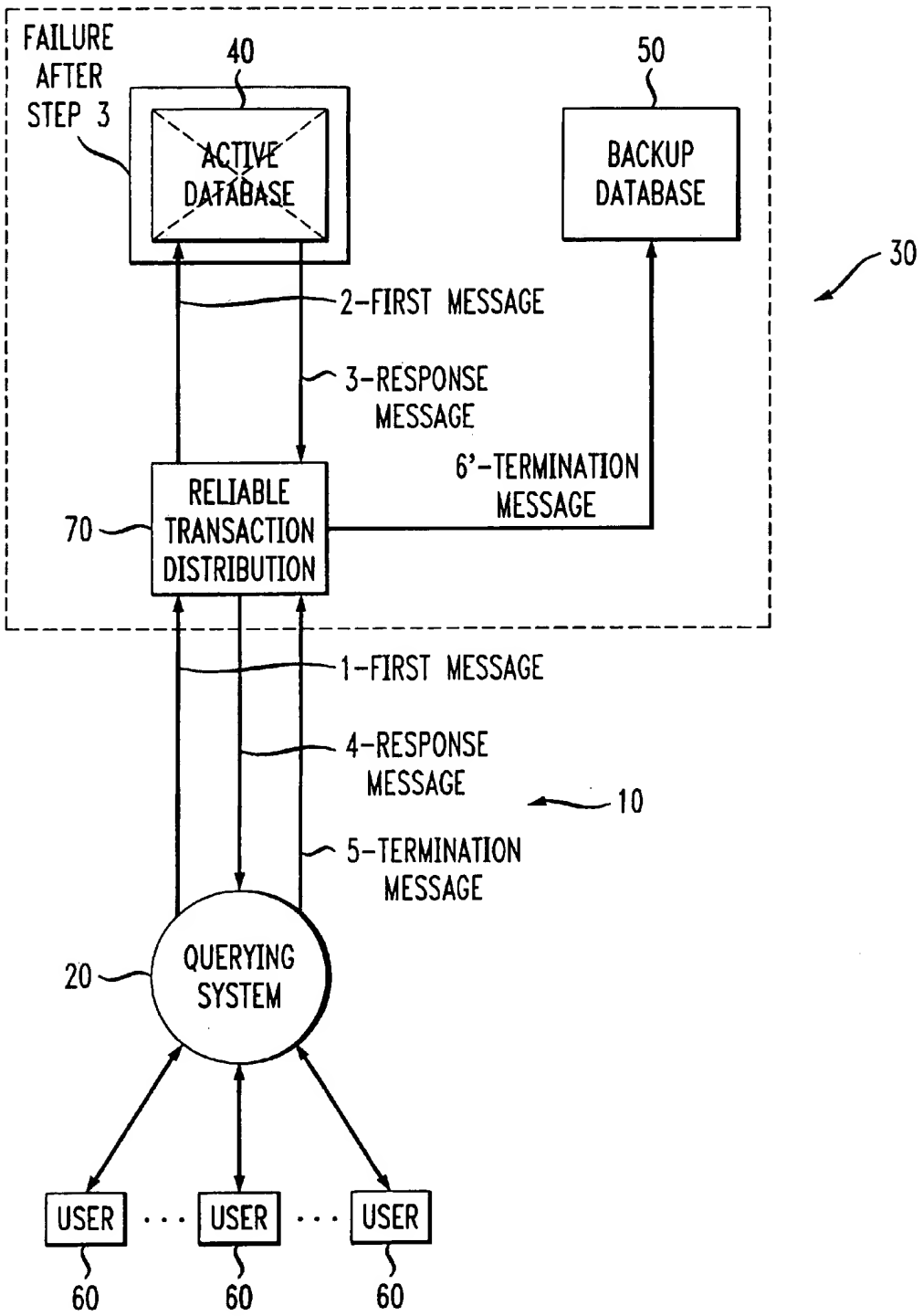


FIG. 3

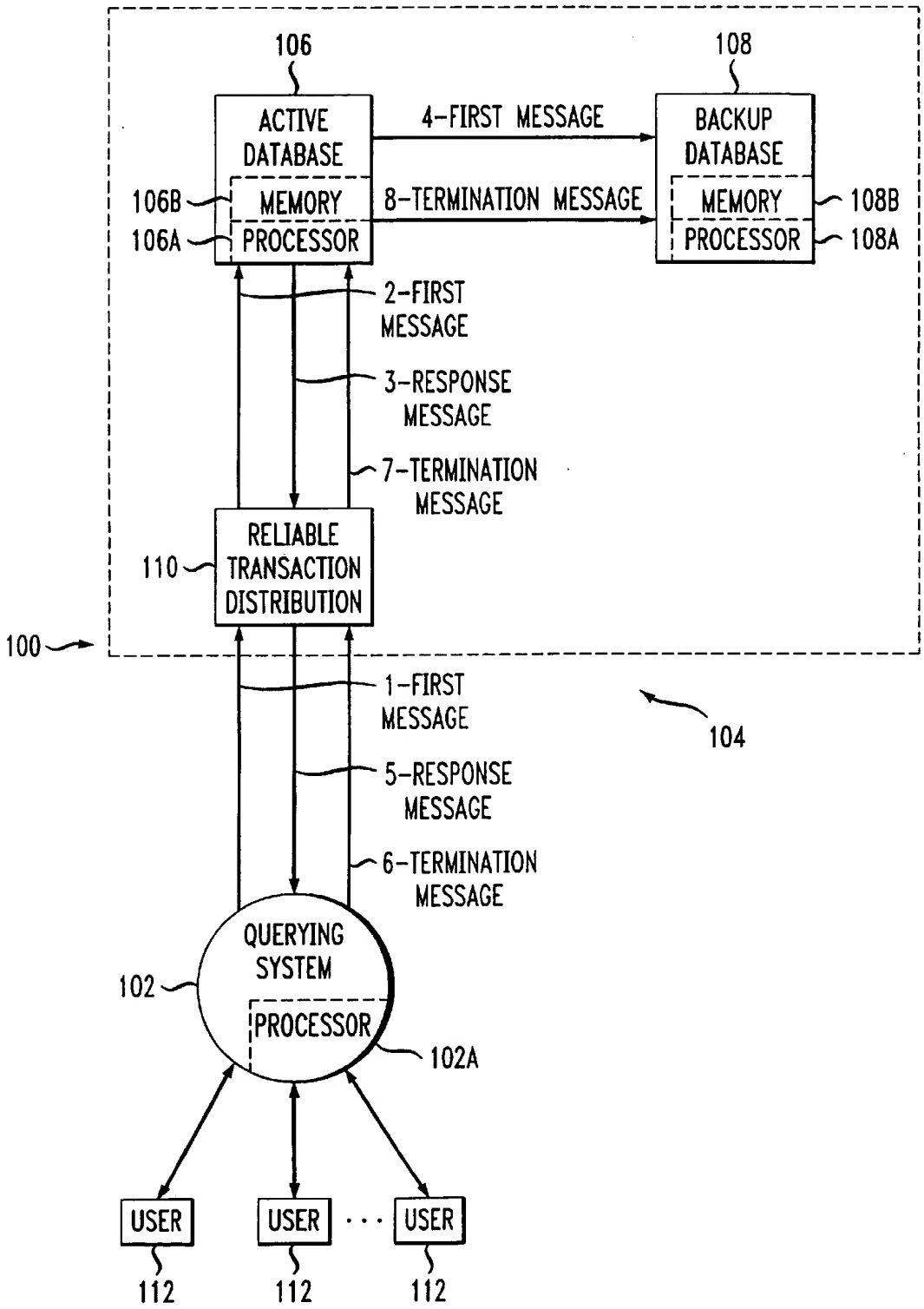
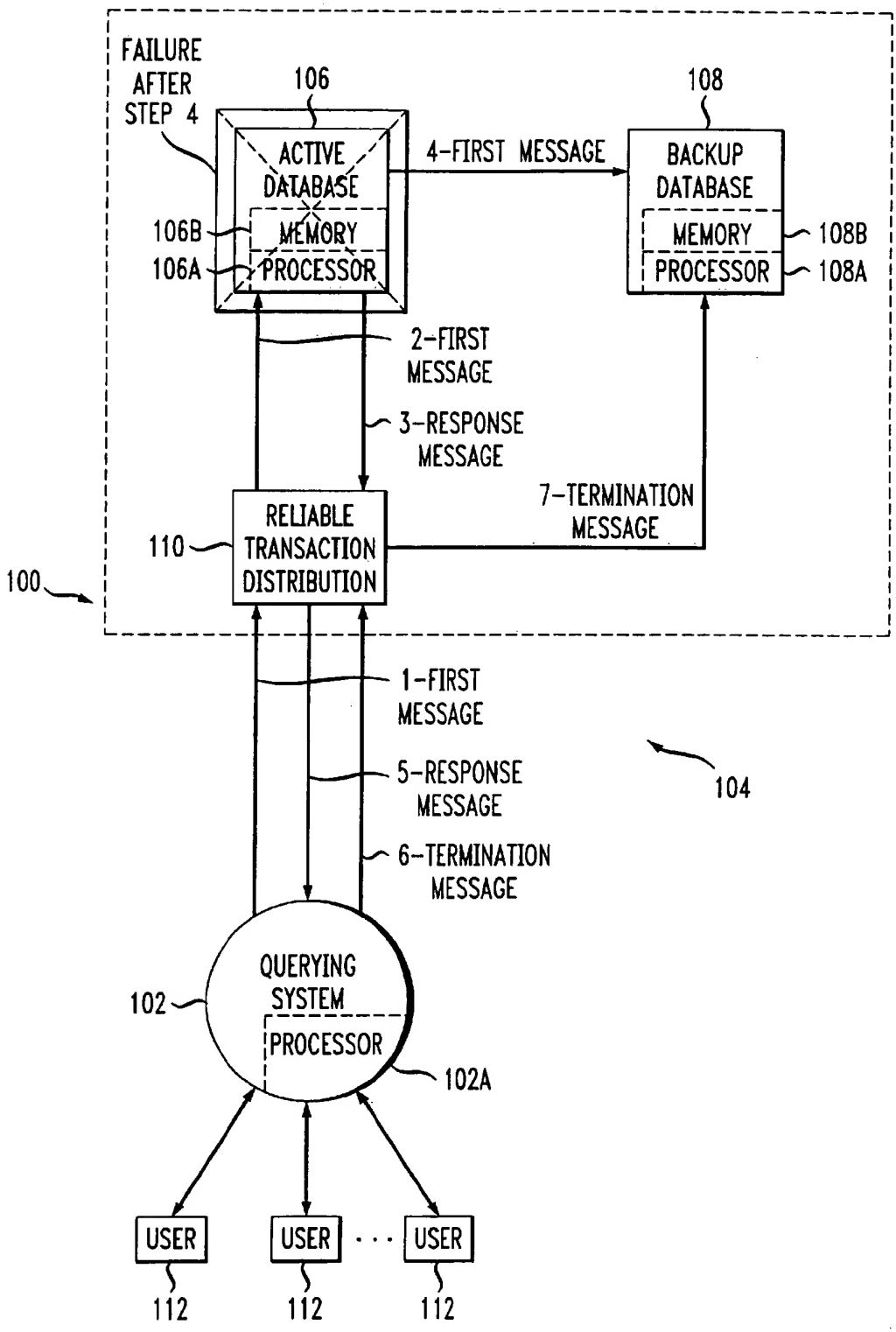

106
108
ACTIVE DATABASE
4-FIRST MESSAGE
BACKUP DATABASE
106B
MEMORY
8-TERMINATION MESSAGE
MEMORY
108B
106A
PROCESSOR
PROCESSOR
108A
2-FIRST MESSAGE
3-RESPONSE MESSAGE
7-TERMINATION MESSAGE
RELIABLE TRANSACTION DISTRIBUTION
110
100
1-FIRST MESSAGE
104
5-RESPONSE MESSAGE
6-TERMINATION MESSAGE
QUERYING SYSTEM
102
PROCESSOR
102A
USER
USER
USER
112
112
112

FIG. 4


FAILURE AFTER STEP 4
106
108
ACTIVE DATABASE
4-FIRST MESSAGE
BACKUP DATABASE
106B
MEMORY
MEMORY
108B
106A
PROCESSOR
PROCESSOR
108A
2-FIRST MESSAGE
3-RESPONSE MESSAGE
7-TERMINATION MESSAGE
RELIABLE TRANSACTION DISTRIBUTION
110
100
1-FIRST MESSAGE
5-RESPONSE MESSAGE
104
6-TERMINATION MESSAGE
QUERYING SYSTEM
102
PROCESSOR
102A
USER
USER
USER
112
112
112

TRANSACTION STATE DATA REPLICATION BY TRANSACTION FORWARDING IN REPLICATED DATABASE SYSTEMS

BACKGROUND OF THE INVENTION

The present invention relates in general to replicated database systems, and, more particularly, to a system and method for improving the reliability of replicated database systems.

Many industry applications, such as airline reservation systems, banking applications and telecommunications, require reliable database transaction processing. A transaction is a sequence of actions or operations that must be performed in its entirety or not at all. Any degradation in reliability has a negative impact on the customer (e.g., the individual making an airline reservation) as well as the service provider (e.g., the airline). For example, if airline transactions are not reliably processed, customers may experience flight overbooking and, as a result, may choose to fly a different airline.

Historically, systems requiring highly reliable transaction processing have been built using highly fault tolerant computing hardware. Typically these systems are guaranteed to function up to 99.9999% of the time which equates to only 30 minutes of downtime/failure time per year. Such a highly fault tolerant computing system is also service reliable as it is nearly always available. Service reliability is a primary objective in designing such computing hardware.

Fault tolerant systems are very expensive and tend to lag behind the "technology innovation curve" by several years. For example, a fault tolerant system may only support on the order of megabytes of memory while a non-fault tolerant system may support on the order of gigabytes of memory. The fault tolerant system providers must transform commercially available hardware into reliability hardened hardware for fault tolerant systems which typically takes two to three years. This transformation process typically entails providing hardware component redundancy and sophisticated hardware failure detection. The underlying objective is to provide a system that rarely fails and is thus nearly always available.

In recent years non-fault tolerant computing systems have become very inexpensive and provide orders of magnitude greater performance in both memory capacities and processor speeds. While the performance of the computing systems have increased, the cost of such computing systems have decreased significantly such that a number of replicated non-fault tolerant computing systems is inexpensive compared to the corresponding fault tolerant computing system. Service reliability may be achieved utilizing less expensive but less reliable computing systems. Instead of utilizing expensive fault tolerant systems, service reliability can be achieved by utilizing redundant non-fault tolerant systems. The idea is to provide an appropriate number of fully replicated systems so that individual system failures do not negatively affect service reliability. Since these systems provide orders of magnitude more performance and capacity, the end result is higher performance and lower cost computing networks with service reliability that is as good as, and often better than, existing networks utilizing fault tolerant systems.

However, there are a number of factors that must be addressed so that a transaction is reliably processed in the midst of system failures. Transaction processing is typically divided into two distinct categories: executing the intent of the transaction and updating/reading the database; and, maintaining the transaction state data required to execute the transaction. There are well understood methods of reliably performing the first category of transaction processing and assuring that data already in a database is stored reliably, e.g., two-phase-commit protocols, redundant array of independent disks (RAID) and shared disk systems.

Transaction state data (TSD) is transient data associated with a single transaction. It is data that is maintained for the life of a transaction to assure that all the parts/messages of the transaction are executed correctly. There is a difference between the data corresponding to the records in a database, e.g., customer specific data, and TSD. Each record in a database includes data unique to that record while TSD is created by the computing system to assure that multiple messages for a particular transaction are correlated appropriately. TSD is also created to store intermediate data needed for subsequent message processing for the transaction.

Referring now to FIG. **1**, a typical replicated database system **10** is illustrated. The replicated database system **10** comprises a querying system **20** and a logical database **30**. The querying system **20** is the system from which all transactions originate. The querying system **20** is configured to generate transactions accessing records in the logical database **30** in response to requests by one of a number of database users **60** accessing the database system **10**. For example, the querying system **20** could be part of an airline reservation system with users accessing and entering reservation data via terminals or a telecommunications switch that is sending transactions to a calling card database for card and personal identification number (PIN) validation. It is the system that requires reliable transaction processing and data storage.

The logical database **30** is the system to which the querying system **20** sends transaction messages for reliable processing and data storage. It is referred to as a logical database because it is actually comprised of multiple physical databases. In the illustrated system, the logical database **30** comprises two physical databases, an active database **40** and a standby database **50**. The logical database **30** also comprises a reliable transaction distributor **70** which receives transaction messages from the querying system **20** and transmits them to one of the databases **40, 50**. The reliable transaction distributor **70** also receives response messages from one of the databases **40, 50** and transmits them to the querying system **20**. It will be appreciated by those skilled in the art that the reliable transaction distributor **70** may be located within the querying system **20**. It will be further appreciated by those skilled in the art that the querying system **20** and the logical database **30** may each comprise a reliable transaction distributor. It will be even further appreciated by those skilled in the art that the reliable transaction distributor **70** is viewed logically as one system but could be comprised of a plurality of physical systems.

The exact number of physical databases within the logical database **30** is completely transparent to the querying system **20**. The reliable transaction distributor **70** is the only component that needs to keep track of the number of physical databases. The databases **40, 50** are fully replicated with each database comprising an identical set of records. The active database **40** is the database that receives transaction messages from the reliable transaction distributor **70** during normal operations, i.e., absent any failures in the active database **40**. The backup database **50** is available for use whenever the active database **40** is experiencing problems.

A typical transaction is described below. Message **1** of the transaction is sent from the querying system **20** to the logical

database **30** for processing (step 1). The reliable transaction database **70** receives message **1** and transmits it to the active database **40** for processing (step 2). The active database **40** retrieves data from an appropriate record in the database and creates internal TSD for subsequent message processing of the transaction. The active database **40** then creates and sends a response message based on the data retrieved from the database and message **1** to the reliable transaction distributor **70** (step 3) for transmission to the querying system **20** (step 4). The querying system **20** then monitors the transaction and transmits a termination message to the logical database **30** (step 5) at the conclusion of the transaction from the view of the querying system **20**. The reliable transaction distributor **70** receives the termination message and transmits it to the active database **40** for processing (step 6). The active database **40** processes the termination message using the TSD. The active database **40** also updates the appropriate records in the database as necessary. The active database **40** transmits the updated data to the backup database **50** (step 7). The backup database **50** updates the appropriate records in the database so that the databases **40, 50** remain fully replicated.

The above illustration could be used in a telecommunication system where a customer uses a pre-paid calling card to place a call. The querying system **20** transmits customer identification data and call originating data in the form of message **1** to the logical database **30** for customer and PIN identification. The active database **40** confirms the customer information from the database along with the balance available on the card. The TSD created by the active database **40** includes message correlation data and call originating data for deriving billing information. The response message instructs the querying system **20** that the customer identification data has been verified and that the call may proceed. The querying system **20** connects the call and monitors the call to determine when the call is completed and disconnected. The querying system **20** determines the duration of the call and transmits this information in the form of the termination message to the logical database **30**. The active database **40** derives a charge for the call and subtracts the charge from the balance in the database. The new balance is then transmitted to the backup database **50** so that the databases **40, 50** remain fully replicated.

Referring now to FIG. **2**, the above transaction will be described assuming that the active database **40** fails after the response message is created and transmitted to the reliable transaction distributor **70** (step 3). The reliable transaction distributor **70** receives the response message and transmits it to the querying system **20** (step 4). Using the calling card example, the call is then connected. The querying system **20** transmits the termination message to the logical database **30** (step 5). Since the active database **40** has failed, the reliable transaction distributor **70** transmits the termination message to the backup database **50** (step 6'). Unfortunately, the backup database **50** does not contain any TSD for the transaction such that message correlation fails and the message is not processed. The transaction itself fails since all of the steps of the transaction could not be completed.

Accordingly, there is a need for a replicated database system and a method for processing a transaction in such a system that allows a transaction to be completed even after the active database fails. There is another need for such a replicated database system that utilizes non-fault tolerant components but maintains service reliability. Preferably, such a system is relatively easy to implement and cost effective.

SUMMARY OF THE INVENTION

The present invention meets these needs by providing a database system comprising a querying system and a logical database having an active database and a backup database. The querying system transmits a message for a transaction to the logical database for processing. The message is transmitted to the active database where the message is processed. The active database creates transaction state data based, in part, on the message. The active database transmits a response message to the querying system and forwards the original message to the backup database. The backup database processes the original message and creates its own transaction state data. The transaction state data in the backup database matches the transaction state data in the active database so that if the active database fails, the backup database includes the requisite transaction state data necessary to complete the transaction.

The querying system processes the response message and transmits a termination message to the logical database for processing. The termination message is transmitted to the active database and processed. The termination message is forwarded to the backup database from the active database and similarly processed. The transaction is then complete and both databases are fully replicated.

According to a first aspect of the present invention, a method for processing a transaction in a replicated database system is provided. The database system comprises a plurality of replicated databases, an active database and at least one backup database. Transaction messages for the transaction are received from a querying system in the active database. The transaction messages are processed for the transaction in the active database. The active database creates response messages based on the transaction messages. The active database also creates active database transaction state data representative of the transaction. The response messages are transmitted to the querying system. The transaction messages for the transaction are forwarded from the active database to the backup database. The transaction messages of the transaction are processed in the backup database with the backup database creating backup database transaction state data representative of the transaction.

The step of processing the transaction messages of the transaction in the backup database may further comprise the step of creating a suppressed response message in the backup database based on the transaction messages. Alternatively, the step of processing the transaction messages of the transaction in the backup database does not comprise the step of creating a response message in the backup database based on the transaction messages forwarded by the active database. The step of forwarding the transaction messages for the transaction from the active database to the backup database may comprise the step of transmitting control header data from the active database to the backup database with the control header data enabling the backup database to process the first message consistent with the active database. The control header data may comprise unique transaction identification data representative of the transaction or time stamp data. The steps of the method may be repeated for a plurality of transactions. The method may further comprise the step of serializing the plurality of transactions so that the plurality of transactions are processed in the same order in both the backup database and the active database.

According to another aspect of the present invention, a method for processing a transaction in a replicated database system is provided. The database system comprises a plurality of replicated databases, an active database and at least one backup database. A first message for the transaction is transmitted from a querying system to the database system.

The first message for the transaction is processed in the active database with the active database creating a response message based on the first message. The active database also creates active database transaction state data representative of the transaction. The response message is transmitted to the querying system. The first message for the transaction is forwarded from the active database to the backup database. The first message for the transaction is processed in the backup database with the backup database creating backup database transaction state data representative of the transaction. A second message for the transaction is transmitted from the querying system to the database system. The second message for the transaction is processed in the active database using the active database transaction state data. The second message for the transaction is forwarded from the active database to the backup database. The second message for the transaction is processed in the backup database using the backup database transaction state data.

The step of processing the first transaction message of the transaction in the backup database may further comprise the step of creating a suppressed response message in the backup database based on the first transaction message. Alternately, the step of processing the first transaction message of the transaction in the backup database does not comprise the step of creating a response message in the backup database based on the first transaction message forwarded by the active database. The step of forwarding the first message for the transaction from the active database to the backup database may further comprise the step of transmitting control header data from the active database to the backup database with the control header data enabling the backup database to process the first message consistent with the active database. The control header data may comprise unique transaction identification data representative of the transaction or time stamp data. The steps of the method may be repeated for a plurality of transactions. The method may further comprise the step of serializing the plurality of transactions so that the plurality of transactions are processed in the same order in the both the backup database and the active database.

The step of processing the first message for the transaction in the active database may comprise the step of storing the active database transaction state data in the active database. The step of processing the first message of the transaction in the backup database may comprise the step of storing the backup database transaction state data in the backup database. The step of processing the first message for the transaction in the active database may comprise the step of accessing at least one of a plurality of records in the active database. The step of processing the first message of the transaction in the backup database may comprise the step of accessing at least one of a plurality of records in the backup database.

According to yet another embodiment of the present invention, a method for processing a transaction in a replicated database system is provided. The database system comprises a plurality of replicated databases, an active database and at least one backup database. A first message for the transaction is transmitted from a querying system to the database system. The first message for the transaction is processed in the active database with the active database creating a response message based on the first message. The active database also creates active database transaction state data representative of the transaction. The response message is transmitted to the querying system while the first message for the transaction is forwarded from the active database to the backup database. The first message of the transaction is processed in the backup database with the backup database creating backup database transaction state data representative of the transaction. A second message for the transaction is transmitted from the querying system to the database system. If the active database is available, then the second message is processed by the active database using the active database transaction state data. The second message for the transaction is also forwarded from the active database to the backup database and processed using the backup database transaction state data. Otherwise, the second message for the transaction is processed by the backup database using the backup database transaction state data.

The step of forwarding the first message for the transaction from the active database to the backup database may comprise the step of transmitting control header data from the active database to the backup database with the control header data enabling the backup database to process the first message consistent with the active database. The control header data may comprise unique transaction identification data representative of the transaction or time stamp data. The steps of the method may be repeated for a plurality of transactions. The method may further comprise the step of serializing the plurality of transactions so that the plurality of transactions are processed in the same order in both the backup database and the active database.

According to a further aspect of the present invention, a transaction processing system comprises a database system and a database querying system. The database system comprises a plurality of replicated databases including an active database and at least one backup database. The active database comprises an active database processor and the backup database comprises a backup database processor. The database querying system comprises a database querying system processor configured to access the database system for processing transactions. The database querying system processor is programmed to transmit a first message for one of the transactions to the database system for processing. The active database processor is programmed to process the first message thereby creating a response message and active database transaction state data representative of the transaction. The active database processor is programmed to transmit the response message to the querying system and to forward the first message to the backup database for processing by the backup database processor. The backup database processor is programmed to process the first message thereby creating backup database transaction state data representative of the transaction. The database querying system is further programmed to transmit a second message for the transaction to the database system for processing. The active database processor is further programmed to process the second message using the active database transaction data, and to forward the second message to the backup database. The backup database processor is further programmed to process the second message using the backup database transaction state data.

The backup processor may be programmed to create a suppressed response message in the backup database based on the first transaction message. Alternatively, the backup processor may be programmed not to create a response message in the backup database based on the first transaction message forwarded by the active database. The active database processor may be further programmed to transmit control header data along with the first message to the backup database so that the backup database processor can process the first message consistent with the active database processor. The control header data may comprise unique transaction identification data representative of the transac-

tion or time stamp data. The database querying system processor, the active database processor and the backup database processor may be programmed to process a plurality of transactions. Preferably, the active database processor is programmed to serialize the plurality of transactions so that the plurality of transactions are processed in the same order in the backup database as in the active database.

Preferably, the active database may comprise an active database memory for storing the active database transaction state data. The backup database may comprise a backup database memory for storing the backup database transaction data. The database system may further comprise a plurality of backup databases. The active database processor may be further programmed to access at least one of a plurality of records in the active database to process the first message. Similarly, the backup database processor may be further programmed to access at least one of a plurality of records in the backup database to process the first message.

According to a still further aspect of the present invention, a transaction processing system comprises a database system and a database querying system. The database system comprises a plurality of replicated databases including an active database and at least one backup database. The active database comprises an active database processor and the backup database comprises a backup database processor. The database querying system comprises a database querying system processor configured to access the database system for processing transactions. The database querying system processor is programmed to transmit a first message for one of the transactions to the database system for processing. The active database processor is programmed to process the first message thereby creating a response message and active database transaction state data representative of the one transaction. The active database processor is further programmed to transmit the response message to the querying system for processing, and to forward the first message for the one transaction to the backup database for processing. The backup database processor is programmed to process the first message thereby creating backup database transaction state data representative of the transaction. The database querying system processor is further programmed to transmit a second message for the one transaction to the database system for processing. If the active database is available, then the second message is processed by the active database processor. The active database processor is further programmed to process the second message using the active database transaction data, and to forward the second message to the backup database. The backup database processor is further programmed to process the second message using the backup database transaction state data. Otherwise, the second message for the transaction is processed by the backup database processor. The backup database processor is programmed to process the second message using the backup database transaction state data.

Accordingly, it is an object of the present invention to provide an improved database system and a method of processing a transaction in such a system that allows a transaction to be completed even after an active database fails. It is another object of the present invention to provide such a replicated database system that utilizes non-fault tolerant components but maintains required service reliability. It is yet another object of the present invention to provide such a system that is relatively easy to implement and cost effective. Other features and advantages of the invention will be apparent from the following description, the accompanying drawings and the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a transaction processing system according to the background art;

FIG. **2** illustrates the transaction processing system of FIG. **1** in a mid-transaction failure mode;

FIG. **3** illustrates a transaction processing system according to the present invention; and

FIG. **4** illustrates the transaction processing system of FIG. **3** in a mid-transaction failure mode.

DETAILED DESCRIPTION OF THE INVENTION

Referring now to FIG. **3**, a transaction processing system **100** according to the present invention is illustrated. The transaction processing system **100** comprises a database querying system **102** and a logical database **104**. The logical database **104** comprises a plurality of replicated databases **106, 108** and a reliable transaction distributor **110**. Each of the plurality of replicated databases **106, 108** include a plurality of substantially identical records. It will be appreciated by those skilled in the art that one or more of the replicated databases **106, 108** may include one or more records that the other database does not include. Each of the records in each of the databases **106, 108** include fields of data. For illustrative purposes only, it is presumed for the following discussion that database **106** is the active database while database **108** is the backup database. It will be appreciated by those skilled in the art that the logical database **104** may comprise a plurality of backup databases.

The database querying system **102** is configured to generate transactions accessing specific records in the logical database **104** in response to requests by one of a number of database users **112** accessing the database system **100**. The database querying system **102** is the system that requires reliable transaction processing and data storage. A transaction is a sequence of actions or operations that must be performed in its entirety or not at all. A typical transaction includes a number of messages transmitted back and forth between the querying system **102** and the logical database **104**. The reliable transaction distributor **110** receives transaction messages from the querying system **102** and transmits them to one of the databases **106, 108** for processing. The reliable transaction distributor **110** also receives response messages from the databases **106, 108** and transmits them to the querying system **102**. It will be appreciated by those skilled in the art that the reliable transaction distributor **110** may be located within the querying system **102**. It will be further appreciated by those skilled in the art that the querying system **102** and the logical database **104** may each comprise a reliable transaction distributor. The actual database accessed by the reliable transaction distributor **110** is transparent to the querying system **102** and the database user because the reliable transaction distributor **110** determines the replicated database to which it sends the transaction message. It will be appreciated by those skilled in the art that the reliable transaction distributor **110** is viewed logically as one system but could be comprised of a plurality of physical systems.

The database querying system **102** includes a database querying system processor **102A** configured to control the databases **106, 108** via the reliable transaction distributor **110**. The active database **106** comprises an active database processor **106A** configured to process messages for a transaction and active database memory **106B** for storing transaction state data. Similarly, the backup database **108** comprises a backup database processor **108A** configured to process messages for a transaction and backup database memory **108B**.

The querying system **102**, the active database **106** and the backup database **108** are configured to process a transaction

as follows. A first message of a first transaction is transmitted from the querying system **102** via the querying system processor **102A** to the logical database **104** for processing (step 1). The reliable transaction distributor **110** receives the first message and transmits it to the active database **106** for processing (step 2). The active database processor **106A** is programmed to process the first message. The active database processor **106A** retrieves data from one or more appropriate records in the active database **106** in response to the first message. It should be apparent that data from additional sources may also be retrieved in response to the first message. The active database processor **106A** then creates active database transaction state data (TSD) based, in part, from the retrieved data and the first message itself. The active database TSD also includes message correlation data so that the future messages of the transaction may be matched with the appropriate TSD. The active database TSD is stored in the active database memory **106B** so as to be available for processing of future messages of the transaction. The active database processor **106A** also creates a response message after processing the first message. The response message is transmitted to the reliable transaction distributor **110** for transmission to the querying system **102** (step 3).

The active database processor **106A** also forwards the first message to the backup database **108** for processing (step 4). Along with the first message, control header data is transmitted to the backup database **108** so that the first message is processed consistent with the active database **106**. The active database **106** and the backup database **108** may not be located in the same room, let alone the same geographic location. Accordingly, the control header data comprises time stamp data, such as date, time of day, day of week or other appropriate time data. The control header also comprises unique transaction identification data so that the backup database **108** and the active database **106** identify each message or transaction consistently. It should be apparent that the control header data may comprise other appropriate data depending on the particular application so that messages are processed consistently by the active database **106** and the backup database **108**.

The backup database processor **108A** is programmed to process the first message in the same manner as the active database **106** such that the backup database processor **108A** retrieves data from one or more appropriate records in the database in response to the first message. It should be apparent that data from additional sources may also be retrieved in response to the first message. The backup database processor **108A** also creates backup database transaction state data (TSD) based, in part, on the retrieved data and the first message itself and its associated Control Header data. The backup database TSD also includes message correlation data so that future messages of the transaction may be matched with the appropriate TSD. The backup database TSD is stored in the backup database memory **108B** so as to be available for processing future messages of the transaction. The active database TSD and the backup database TSD should be operationally identical so that either database **106**, **108** can process future messages for the transaction. While the active database processor **106A** is programmed so that it does create a response message, the backup database processor **108A** is programmed to not create a response message. If a second response message was created and transmitted to the querying system **102**, the querying system **102** would attempt to process both response messages leading to duplicate processing of a single transaction. Alternatively, it should be apparent that the backup database processor **108A** may be programmed to create a suppressed response message. That is, the backup database processor **108A** performs the steps required to create the response message but then suppresses the response message as long as it is in a backup mode of operation so that multiple response messages are not transmitted in the system.

The TSD in the active database **106** is replicated by the backup database **108** by virtue of the first message being separately processed by the backup database **108**. The TSD in the backup database **108** is therefore available in the event that the active database **106** experiences a failure. As the TSD can be many kilobytes in size, it is not practical to simply send the TSD from the active database **106** to the backup database **108** every time that it changes during processing of a transaction. Accordingly, the TSD is replicated by having each message that is processed by the active database **106** also processed by the backup database **108**.

The reliable transaction distributor **110** receives the response message from the active database **106** and transmits it to the querying system **102** where it is processed by the querying system processor **102A** (step 5). Once the transaction is complete from the perspective of the querying system **102**, the querying system processor **102A** transmits a second or termination message to the logical database **104** for processing (step 6). Depending on the application, the transaction may require one or more additional messages to be transmitted to the logical database **104** for processing prior to the transmission of the termination message.

The reliable transaction distributor **110** receives the termination message and transmits it to the active database **106** for processing (step 7). The active database processor **106A** is programmed to process the termination message using the active database TSD stored in the memory **106B**. The active database processor **106A** also updates the appropriate records in the database, as necessary, using the active database TSD and the contents of the termination message. The active database processor **106A** forwards the termination message along with appropriate control header data to the backup database **108** for processing (step 8). The backup database processor **108A** processes the termination message the same as the active database **106** except that the backup database TSD stored in the memory **108B** and control header data are used. The backup database processor **108A** updates the corresponding appropriate records in the backup database **108**, as necessary, using the backup database TSD and the contents of the termination message. The appropriate record in the backup database **108** is modified/updated by virtue of the termination message being processed by the backup database **108**. The backup database **108** and the active database **106** therefore remain fully replicated without the active database **106** having to transmit data to the backup database **108** to modify/update records. Once the termination message has been processed, the TSD in the active database memory **106B** and the backup database memory **108B** are deleted.

Referring now to FIG. **4**, the above transaction will now be described assuming that the active database **106** fails after the first message is forwarded to the backup database **108** (step 4). The reliable transaction distributor **110** receives the response message from the active database **106** and transmits it to the querying system **102** where it is processed by the querying system processor **102A** (step 5). Once the transaction is complete from the perspective of the querying system **102**, the querying system processor **102A** transmits a termination message to the logical database **104** for processing (step 6).

The reliable transaction distributor **110** receives the termination message and transmits it to the backup database

108 for processing (step 7') since the active database 106 is unavailable. The backup database processor 108A processes the termination message using the backup database TSD. The backup database processor 108A updates the appropriate records in the database as necessary using the backup database TSD and the contents of the termination message. Since the backup database 108 includes its own internal TSD, message correlation succeeds and the message is processed successfully. Further, since every message of the transaction is processed successfully, the transaction is processed successfully. Accordingly, forwarding transactions from the active database 106 to the backup database 108 for separate processing enables non-fault tolerant computing systems to be used to process transactions reliably.

In some applications, the transaction messages processed by the backup database 108 need to be processed in the same order as processed in the active database 106 to maintain consistent TSD. Serialization of the transaction messages is complicated in that there are often multiple transaction processors within a database. One known method of serializing transaction messages is accomplished using a data structure having a double-linked list (DLL) of transaction messages. Messages requiring replication by the active database 106 have a replication indicator data field set "active" upon completion of message processing. These messages "bubble" to the top-of-queue. When such messages reach the top of the queue, the transaction message and corresponding control header data are transmitted to the backup database 108. The message is then cleared from the queue and the replication indicator data field reset.

Transaction messages are thus distributed to the transaction processors in a first-in-first-out (FIFO) manner. For example, assume that the active database 108 includes four transaction processors, TP1, TP2, TP3 and TP4. Message A is sent to TP 1, message B is sent to TP 2, message C is sent to TP 3 and message D is sent to TP 4. Though message D is the last of the messages to be distributed to a TP, it could be the first message completely processed. If the message was simply sent to the backup database 108 upon successful processing by the active database 106, message order would be lost and the potential for TSD inconsistencies would increase. Therefore, a data structure such as the DLL is used to assure proper serialization and sequencing to the backup database 108. Messages that require replication, such as message D in the example, will "bubble" to the top of the queue in the order in which they were received and not in the order in which they were completed. The messages are thus serialized for proper and consistent processing in both the active database 106 and the backup database 108. It will be appreciated by those skilled in the art that some applications will not require exact ordering of messages.

Having described the invention in detail and by reference to preferred embodiments thereof, it will be apparent that modifications and variations are possible without departing from the scope of the invention defined in the appended claims.

What is claimed is:

1. A method for processing a transaction in a replicated database system comprising a plurality of replicated databases, said plurality of replicated databases comprising an active database and at least one backup database, said transaction comprising a plurality of sequential messages that must be performed to complete said transaction, said method comprising the steps of:

receiving at least one message for said transaction from a querying system in said active database;

processing said at least one message for said transaction in said active database, said active database creating corresponding response messages based on said at least one message and creating transaction state data for said active database so that said active database can process subsequently received messages of said transaction;

transmitting said response messages to said querying system;

forwarding said at least one message for said transaction and control header data from said active database to said at least one backup database, said control header data enabling said backup database to process said at least one message consistent with said active database; and

processing said at least one message of said transaction in said at least one backup database, said at least one backup database creating transaction state data for said backup database so that said backup database can directly process subsequent messages of said transaction in the event of failure of said active database.

2. The method of claim 1, wherein said step of processing said at least one message for said transaction in said at least one backup database further comprises the step of creating a suppressed response message in said at least one backup database based on said at least one message.

3. The method of claim 1, wherein said step of processing said at least one message for said transaction in said at least one backup database does not comprise the step of creating a response message in said at least one backup database based on said at least one message forwarded by said active database.

4. The method of claim 1, wherein said control header data comprises unique transaction identification data representative of said transaction.

5. The method of claim 1, wherein said control header data comprises time stamp data.

6. The method of claim 1, wherein said steps are repeated for a plurality of transactions.

7. The method of claim 6, further comprising the step of serializing said plurality of transactions so that said plurality of transactions are processed in the same order in said at least one backup database as in said active database.

8. A method for processing a transaction in a replicated database system comprising a plurality of replicated databases, said plurality of replicated databases comprising an active database and at least one backup database, said transaction comprising a plurality of sequential messages that must be performed to complete said transaction, said method comprising the steps of:

transmitting a first message for said transaction from a querying system to said replicated database system;

processing said first message for said transaction in said active database, said active database creating a response message based on said first message and creating transaction state data for said active database so that said active database can process subsequently received messages of said transaction;

transmitting said response message to said querying system;

forwarding said first message for said transaction and control header data from said active database to said at least one backup database, said control header data enabling said backup database to process said first message consistent with said active database;

processing said first message for said transaction in said at least one backup database, said at least one backup database creating transaction state data for said backup database so that said backup database can process subsequent messages of said transaction;

transmitting a second message for said transaction from said querying system to said database system;

processing said second message for said transaction in said active database using said active database transaction state data;

forwarding said second message for said transaction and control header data from said active database to said at least one backup database, said control header data enabling said backup database to process said second message consistent with said active database; and

processing said second message for said transaction in said at least one backup database using said backup database transaction state data.

9. The method of claim **8**, wherein said step of processing said first transaction message of said transaction in said at least one backup database further comprises the step of creating a suppressed response message in said at least one backup database based on said first transaction message.

10. The method of claim **8**, wherein said step of processing said first transaction message of said transaction in said at least one backup database does not comprise the step of creating a response message in said at least one backup database based on said first transaction message forwarded by said active database.

11. The method of claim **8**, wherein said control header data comprises unique transaction identification data representative of said transaction.

12. The method of claim **8**, wherein said control header data comprises time stamp data.

13. The method of claim **8**, wherein said steps are repeated for a plurality of transactions.

14. The method of claim **13**, further comprising the step of serializing said plurality of transactions so that said plurality of transactions are processed in the same order in said at least one backup database as in said active database.

15. The method of claim **8**, wherein said step of processing said first message for said transaction in said active database comprises the step of storing said active database transaction state data in said active database.

16. The method of claim **8**, wherein said step of processing said first message of said transaction in said at least one backup database comprises the step of storing said backup database transaction state data in said at least one backup database.

17. The method of claim **8**, wherein said step of processing said first message for said transaction in said active database comprises the step of accessing at least one of a plurality of records in said active database.

18. The method of claim **17**, wherein said step of processing said first message of said transaction in said at least one backup database comprises the step of accessing at least one of a plurality of records in said at least one backup database.

19. A method for processing a transaction in a replicated database system Comprising a plurality of replicated databases, said plurality of replicated databases comprising an active database and at least one backup database, said transaction comprising a plurality of sequential messages that must be performed to complete said transaction, said method comprising the steps of:

transmitting a first message for said transaction from a querying system to said database system;

processing said first message for said transaction in said active database, said active database creating a response message based on said first message and creating transaction state data for said active database so that said active database can process subsequently received messages of said transaction;

transmitting said response message to said querying system;

forwarding said first message for said transaction and control header data from said active database to said at least one backup database, said control header data enabling said backup database to process said first message consistent with said active database;

processing said first message of said transaction in said at least one backup database, said at least one backup database creating transaction state data for said backup database so that said backup database can process subsequent messages of said transaction;

transmitting a second message for said transaction from said querying system to said database system;

if said active database is available, then:

processing said second message for said transaction in said active database using said active database transaction state data;

forwarding said second message for said transaction and control header data from said active database to said at least one backup database, said control header data enabling said backup database to process said second message consistent with said active database; and

processing said second message for said transaction in said at least one backup database using said backup database transaction state data; and if said active database is not available:

processing said second message for said transaction in said at least one backup database using said backup database transaction state data.

20. The method of claim **19**, wherein said control header data comprises unique transaction identification data representative of said transaction.

21. The method of claim **19**, wherein said control header comprises time stamp data.

22. The method of claim **19**, wherein said steps are repeated for a plurality of transactions.

23. The method of claim **22**, further comprising the step of serializing said plurality of transactions so that said plurality of transactions are processed in the same order in said at least one backup database as in said active database.

24. A system for processing a transaction comprising a sequence of messages that must be performed to complete said transaction, said system comprising:

a database system comprising a plurality of replicated databases, said plurality of replicated databases comprising an active database and at least one backup database, said active database comprising an active database processor and said at least one backup database comprising a backup database processor; and

a database querying system comprising a database querying system processor configured to access said database system for processing transactions, said database querying system processor programmed to transmit a first message for one of said transactions to said database system for processing by said active database processor;

wherein said active database processor is programmed to:

update at least one record in said data storage unit of said active database process in response to said first message and create a response message and transaction state data for said active database so that said active database can process messages subsequent to said first message of said one of said transactions;

transmit said response message to said querying system for processing by said querying system processor; and

forward said first message for said one of said transactions and control header data to said at least one backup database for processing by said at least one backup database processor, said control header data enabling said backup database to process said first message consistent with said active database;

and wherein said at least one backup database processor is programmed to process said first message to create transaction state data for completion of said one of said transactions;

and wherein said database querying system is further programmed to transmit a second message for said one transaction to said database system;

and wherein said active database processor is further programmed to:

process said second message using said active database transaction data; and

forward said second message and control header data to said at least one backup database, said control header data enabling said backup database to process said second message consistent with said active database;

and wherein said at least one backup database processor is programmed to process said second message using said backup database transaction state data.

25. The system of claim **24**, wherein said at least one backup database processor is programmed to create a suppressed response message in said at least one backup database based on said first transaction message.

26. The system of claim **24**, wherein said at least one backup database processor is programmed not to create a response message in said at least one backup database based on said first transaction message forwarded by said active database.

27. The system of claim **24**, wherein said control header data comprises unique transaction identification data representative of said transaction.

28. The system of claim **24**, wherein said control header data comprises time stamp data.

29. The system of claim **24**, wherein said database querying system processor, said active database processor and said at least one backup database processor are programmed to process a plurality of transactions.

30. The system of claim **29**, wherein said active database processor is programmed to serialize said plurality of transactions so that said plurality of transactions are processed in the same order in said at least one backup database as in said active database.

31. The system of claim **24**, wherein said active database comprises an active database memory for storing said active database transaction state data.

32. The system of claim **24**, wherein said at least one backup database comprises a backup database memory for storing said backup database transaction data.

33. The system of claim **24**, wherein said database system comprises a plurality of backup databases.

34. The system of claim **24**, wherein said active database processor is further programmed to access at least one of a plurality of records in said active database to process said first message.

35. The system of claim **34**, wherein said at least one backup database processor is further programmed to access at least one of a plurality of records in said at least one backup database to process said first message.

36. A system for processing a transaction comprising a sequence of operations that must be performed to complete said transaction, said system comprising:

a replicated database system comprising a reliable transaction distributor and a plurality of replicated databases, said reliable transaction distributor distributing messages controlling said operations to said plurality of replicated databases which comprise an active database and at least one backup database, said active database comprising an active database processor and said at lease one backup database comprising a backup database processor; and

a database querying system comprising a database querying system processor configured to access said plurality of replicated databases for processing transactions, said database querying system processor programmed to transmit a first message for one of said transactions to said database system;

wherein said active database processor is programmed to:

process said first message thereby creating a response message and transaction state data for said active database so that said active database can process subsequent messages of said one transaction;

transmit said response message to said querying system for processing by said querying system processor: and

forward said first message for said one transaction and control header data to said at least one backup database for processing by said backup database processor, said control header data enabling said backup database to process said first message consistent with said active database;

and wherein said at least one backup database processor is programmed to process said first message thereby creating transaction state data for said backup database so that said backup database can process subsequent messages of said one transaction;

and wherein said database querying system processor is further programmed to transmit a second message for said one transaction to said database system;

and wherein if said active database is available then said active database processor is further programmed to:

process said second message using said active database transaction data; and

forward said second message and control header data to said at least one backup database, said control header data enabling said backup database to process said second message consistent with said active database;

and wherein said at least one backup database processor is programmed to process said second message using said backup database transaction state data;

and if said active database is not available:

said at least one backup database processor is programmed to process said second message received directly from said reliable transaction distributor using said backup database transaction state data.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,347,322 B1
DATED : February 12, 2002
INVENTOR(S) : Robert L. Bogantz, Gary Michael Green and Sidney Dean Hester

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 3,
Line 54, "step 6'" should read -- step 6 --.

Column 5,
Line 39, "in the both" should read -- in both --.

Column 4,
Line 15, new paragraph starts here but should not be a separate paragraph --.

Column 8,
Lines 15 and 20, "108 include" should read -- 108 includes --.

Column 11,
Line 1, "step 7' " should read -- step 7 --.

Column 13,
Line 55, "Comprising" should read -- comprising --.

Column 16,
Line 29, "processor:" should read -- processor; --.

Signed and Sealed this

Ninth Day of July, 2002

Attest:

Attesting Officer

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006085298A

United States Patent [19]

Ohran

[11] Patent Number: 6,085,298

[45] Date of Patent: Jul. 4, 2000

[54] **COMPARING MASS STORAGE DEVICES THROUGH DIGESTS THAT ARE REPRESENTATIVE OF STORED DATA IN ORDER TO MINIMIZE DATA TRANSFER**

[75] Inventor: **Richard Ohran**, Provo, Utah

[73] Assignee: **Vinca Corporation**, Orem, Utah

[21] Appl. No.: **09/165,180**

[22] Filed: **Oct. 1, 1998**

Related U.S. Application Data

[60] Division of application No. 08/747,151, Nov. 8, 1996, Pat. No. 5,835,953, which is a continuation-in-part of application No. 08/322,697, Oct. 13, 1994, Pat. No. 5,649,152.

[51] **Int. Cl.**[7] **G06F 12/16**

[52] **U.S. Cl.** **711/162**; 711/154; 711/161

[58] **Field of Search** 711/162, 154, 711/161, 167; 395/182.04, 182.03, 182.05; 714/118, 13

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,349,655	9/1994	Mann	395/182.04
5,649,196	7/1997	Woodhill et al.	707/204
5,719,889	2/1998	Iadanza	395/182.04
5,819,020	10/1998	Beeler, Jr.	395/182.03

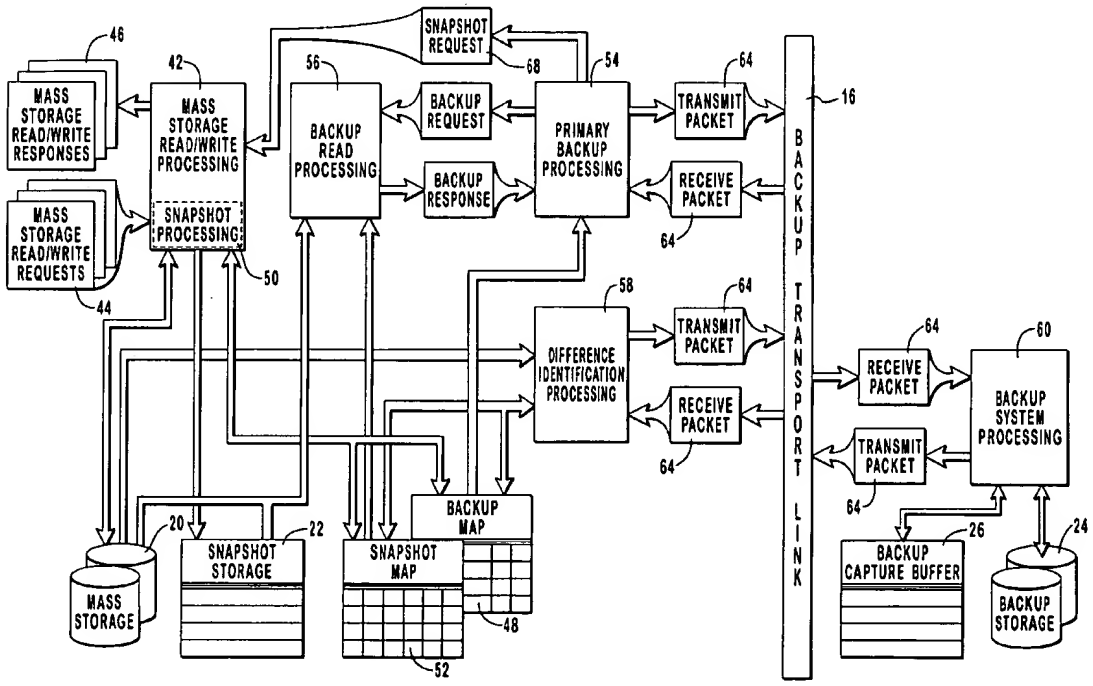
Primary Examiner—Tuan V. Thai

Attorney, Agent, or Firm—Workman, Nydegger, Seeley

[57] **ABSTRACT**

A system and method for comparing mass storage devices. Generally, a mass storage device is subdivided into data blocks representing physical storage locations of some particular size. For each one (or group) of data block(s), a digest is calculated, wherein the digest is an alternate representation of the data stored within the data block(s). A digest is highly dependent on the data it represents such that different data is extremely likely to result in different digests. This high dependence allows for comparing digests rather than directly comparing the data stored at each data block. Furthermore, by calculating a digest that requires fewer bits than the data block(s) it represents, the amount of data that must be transferred between mass storage devices during comparison is minimized.

28 Claims, 11 Drawing Sheets



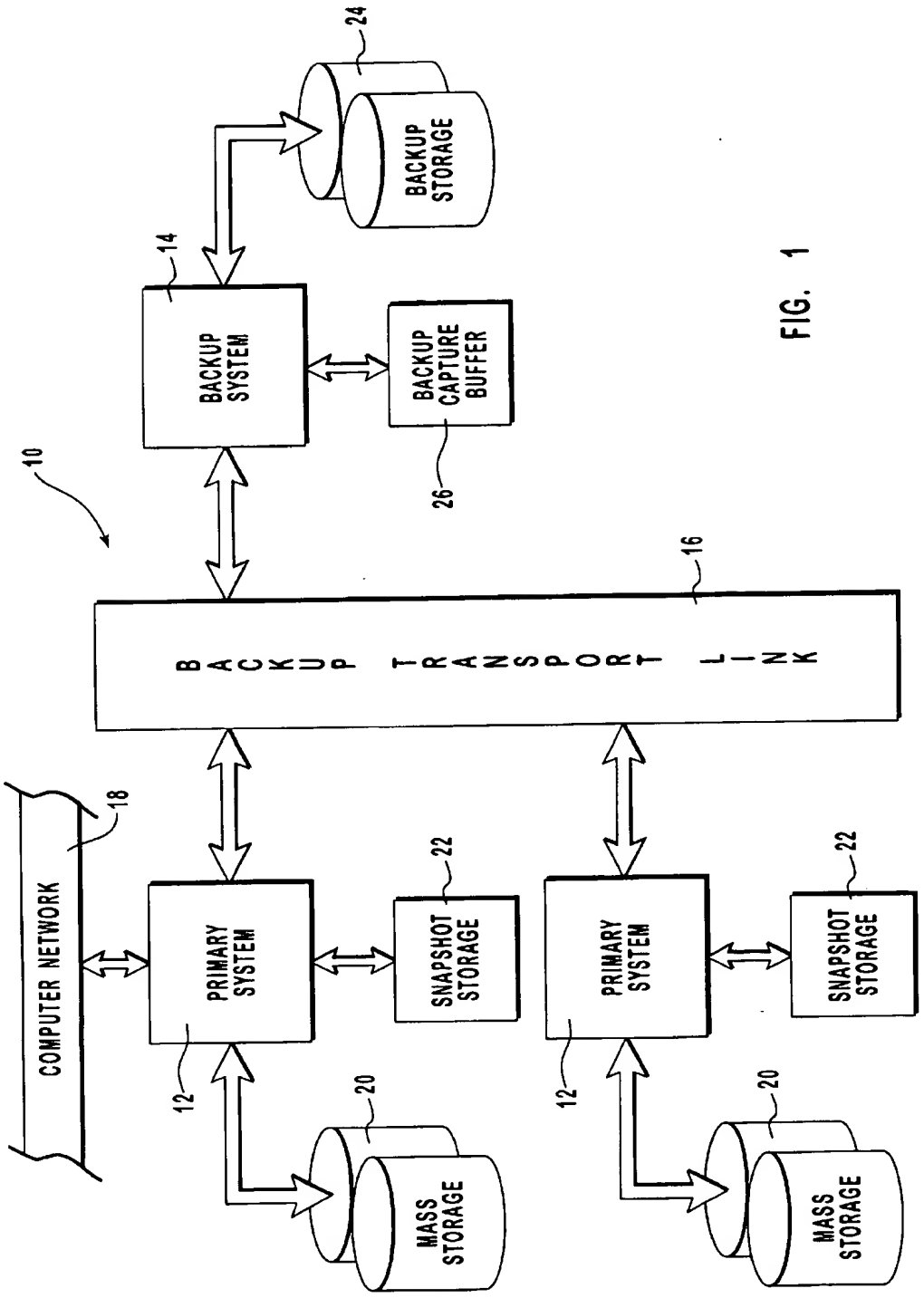

24
BACKUP STORAGE
14
BACKUP SYSTEM
BACKUP CAPTURE BUFFER
26
10
16
BACKUP TRANSPORT LINK
18
COMPUTER NETWORK
PRIMARY SYSTEM
22
SNAPSHOT STORAGE
12
20
MASS STORAGE

FIG. 1

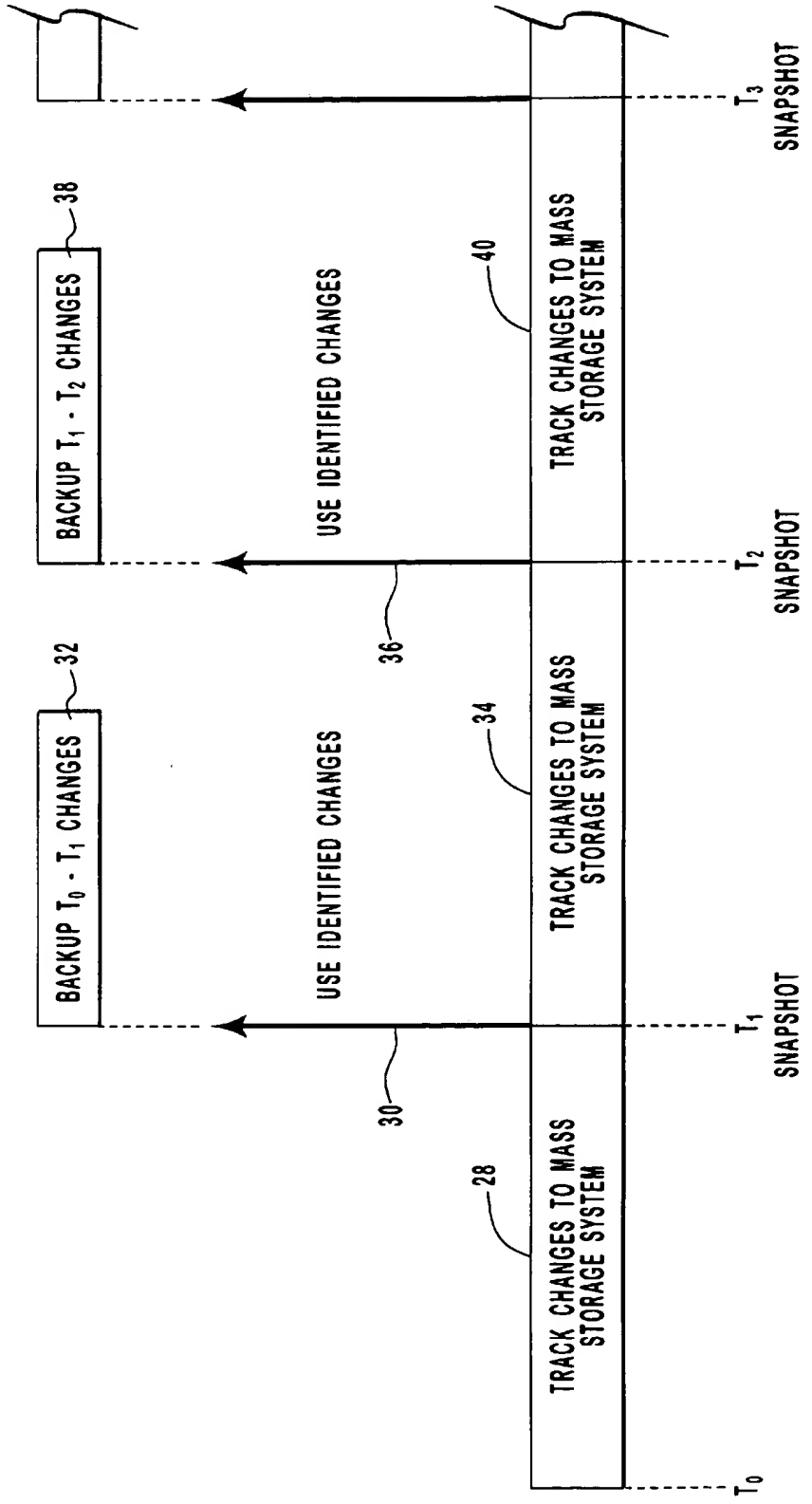

BACKUP T_0 - T_1 CHANGES
32
USE IDENTIFIED CHANGES
30
TRACK CHANGES TO MASS STORAGE SYSTEM
28
34
T_1 SNAPSHOT
BACKUP T_1 - T_2 CHANGES
38
USE IDENTIFIED CHANGES
36
TRACK CHANGES TO MASS STORAGE SYSTEM
40
T_2 SNAPSHOT
T_3 SNAPSHOT
T_0

FIG. 2

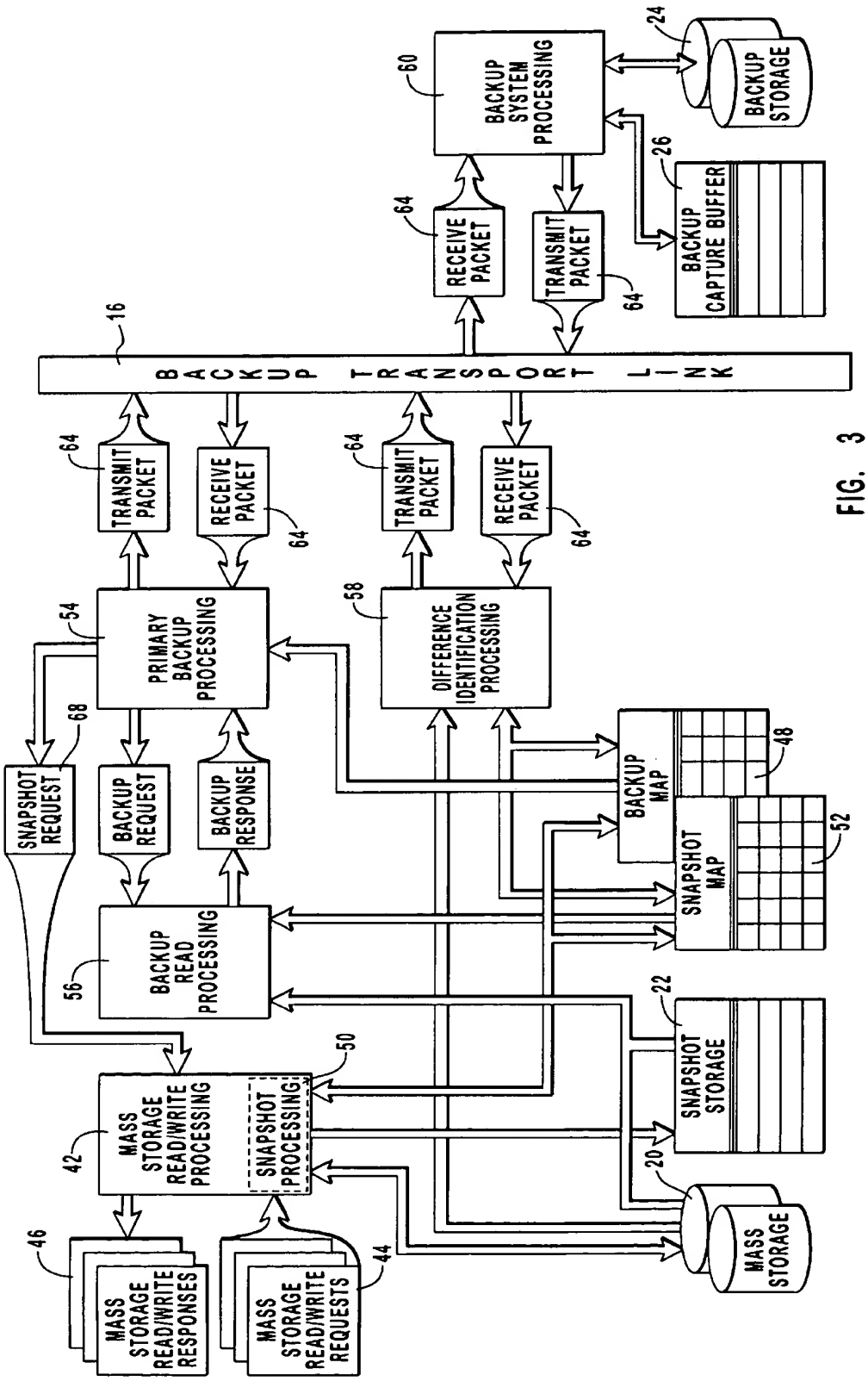

BACKUP SYSTEM PROCESSING
60
BACKUP STORAGE
24
BACKUP CAPTURE BUFFER
26
RECEIVE PACKET
64
TRANSMIT PACKET
64
16
BACKUP TRANSPORT LINK
TRANSMIT PACKET
64
RECEIVE PACKET
64
TRANSMIT PACKET
64
RECEIVE PACKET
64
PRIMARY BACKUP PROCESSING
54
DIFFERENCE IDENTIFICATION PROCESSING
58
SNAPSHOT REQUEST
68
BACKUP REQUEST
BACKUP RESPONSE
BACKUP MAP
48
SNAPSHOT MAP
52
BACKUP READ PROCESSING
56
SNAPSHOT STORAGE
22
MASS STORAGE READ/WRITE PROCESSING
SNAPSHOT PROCESSING
50
42
20
MASS STORAGE
MASS STORAGE READ/WRITE RESPONSES
46
MASS STORAGE READ/WRITE REQUESTS
44

FIG. 3

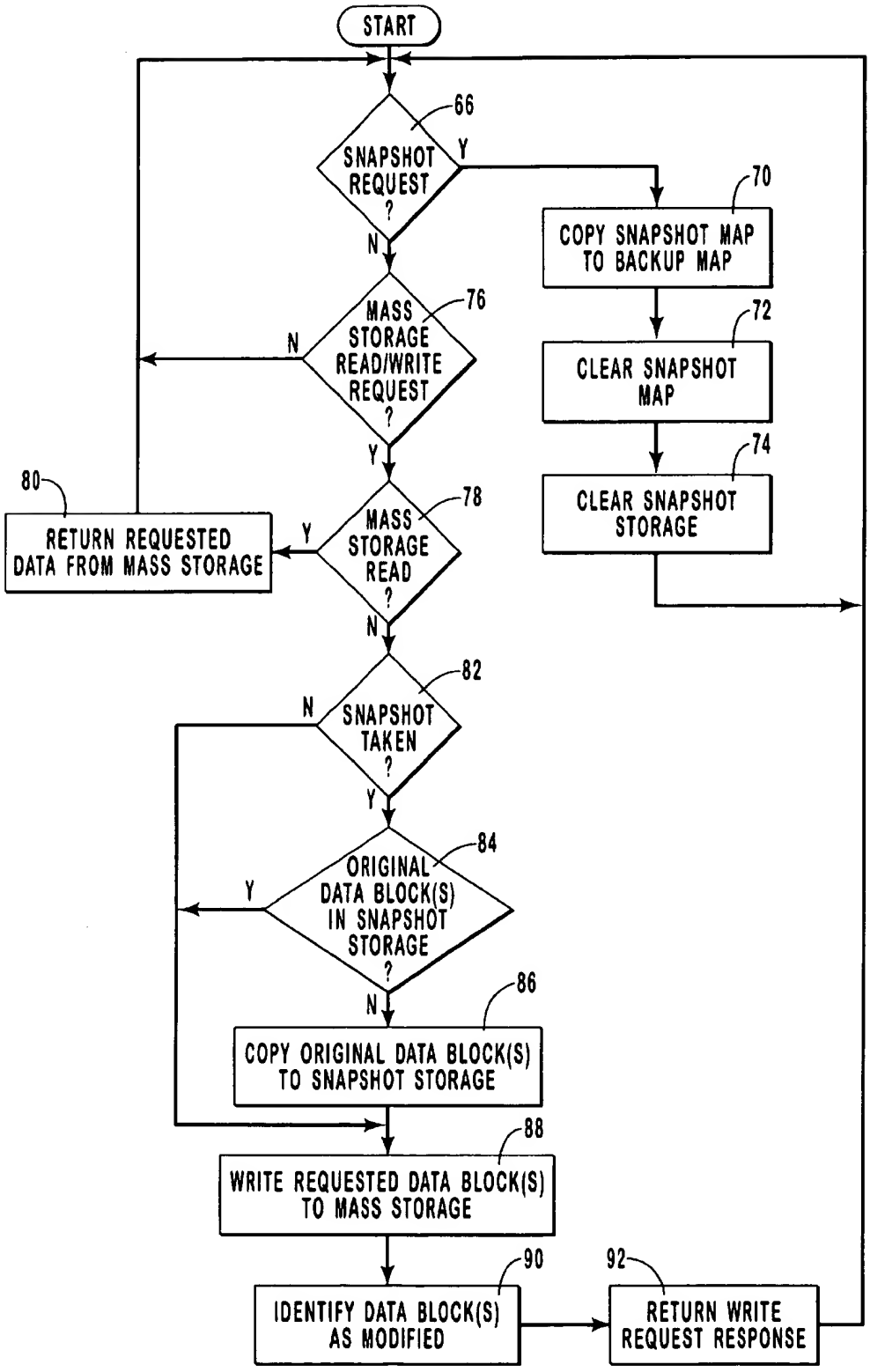

START
66
SNAPSHOT REQUEST ?
Y
N
70
COPY SNAPSHOT MAP TO BACKUP MAP
72
CLEAR SNAPSHOT MAP
74
CLEAR SNAPSHOT STORAGE
76
MASS STORAGE READ/WRITE REQUEST ?
N
Y
80
RETURN REQUESTED DATA FROM MASS STORAGE
Y
78
MASS STORAGE READ ?
N
82
SNAPSHOT TAKEN ?
N
Y
84
ORIGINAL DATA BLOCK(S) IN SNAPSHOT STORAGE ?
Y
N
86
COPY ORIGINAL DATA BLOCK(S) TO SNAPSHOT STORAGE
88
WRITE REQUESTED DATA BLOCK(S) TO MASS STORAGE
90
IDENTIFY DATA BLOCK(S) AS MODIFIED
92
RETURN WRITE REQUEST RESPONSE

FIG. 4

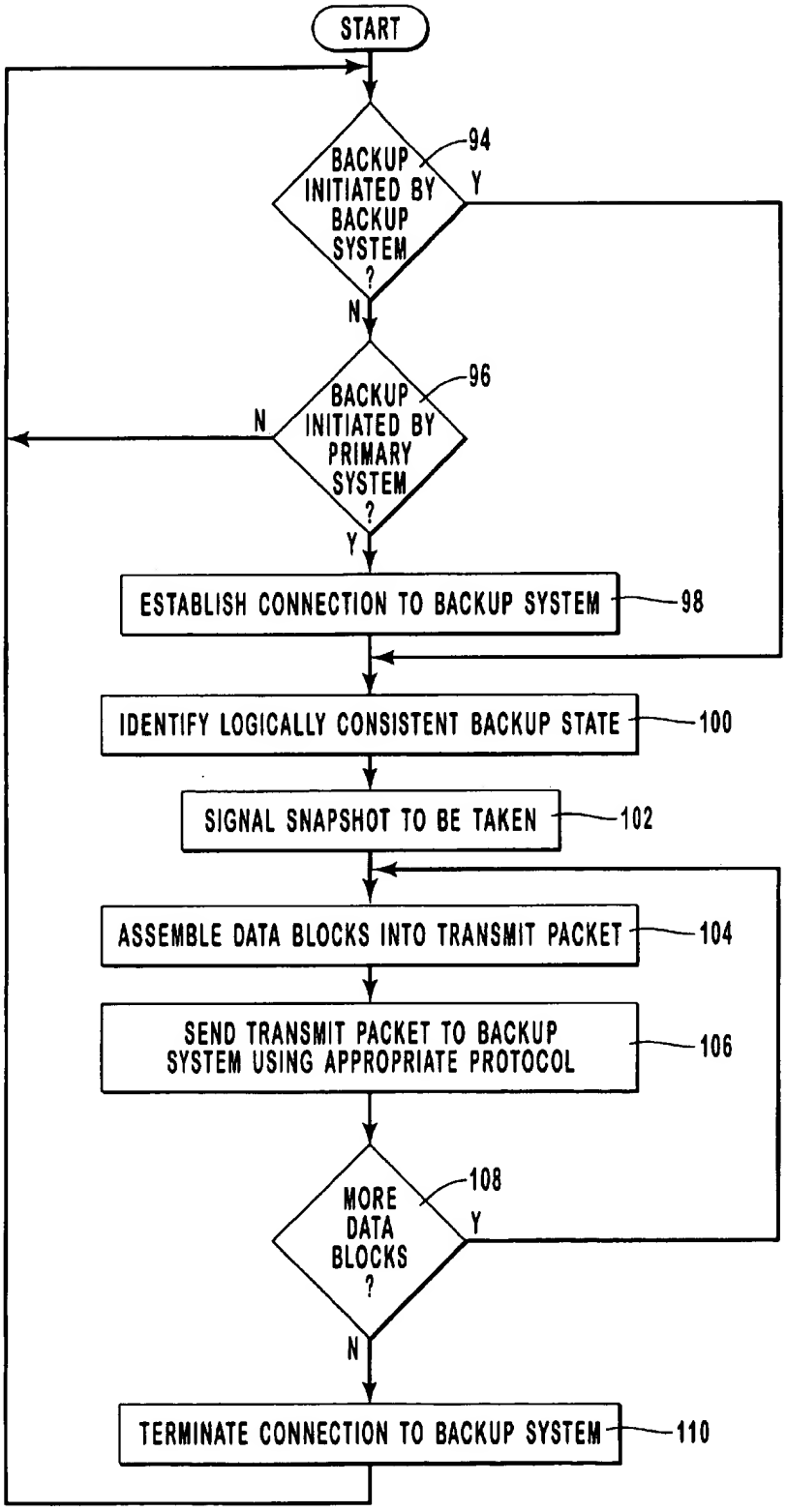

START
BACKUP INITIATED BY BACKUP SYSTEM ?
94
Y
N
BACKUP INITIATED BY PRIMARY SYSTEM ?
96
N
Y
ESTABLISH CONNECTION TO BACKUP SYSTEM
98
IDENTIFY LOGICALLY CONSISTENT BACKUP STATE
100
SIGNAL SNAPSHOT TO BE TAKEN
102
ASSEMBLE DATA BLOCKS INTO TRANSMIT PACKET
104
SEND TRANSMIT PACKET TO BACKUP SYSTEM USING APPROPRIATE PROTOCOL
106
MORE DATA BLOCKS ?
108
Y
N
TERMINATE CONNECTION TO BACKUP SYSTEM
110

FIG. 5

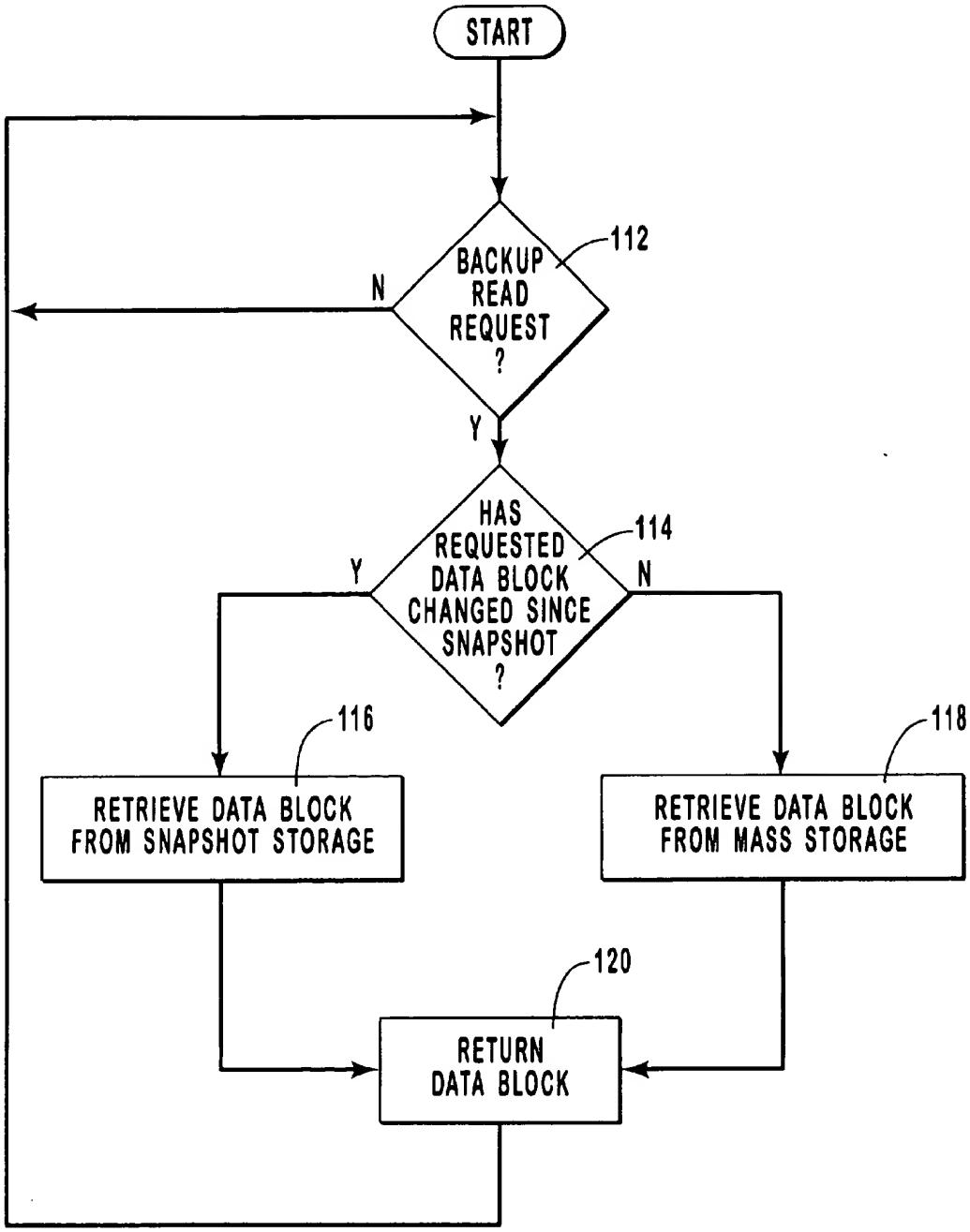

START
BACKUP READ REQUEST ?
112
N
Y
HAS REQUESTED DATA BLOCK CHANGED SINCE SNAPSHOT ?
114
Y
N
116
RETRIEVE DATA BLOCK FROM SNAPSHOT STORAGE
118
RETRIEVE DATA BLOCK FROM MASS STORAGE
120
RETURN DATA BLOCK

FIG. 6

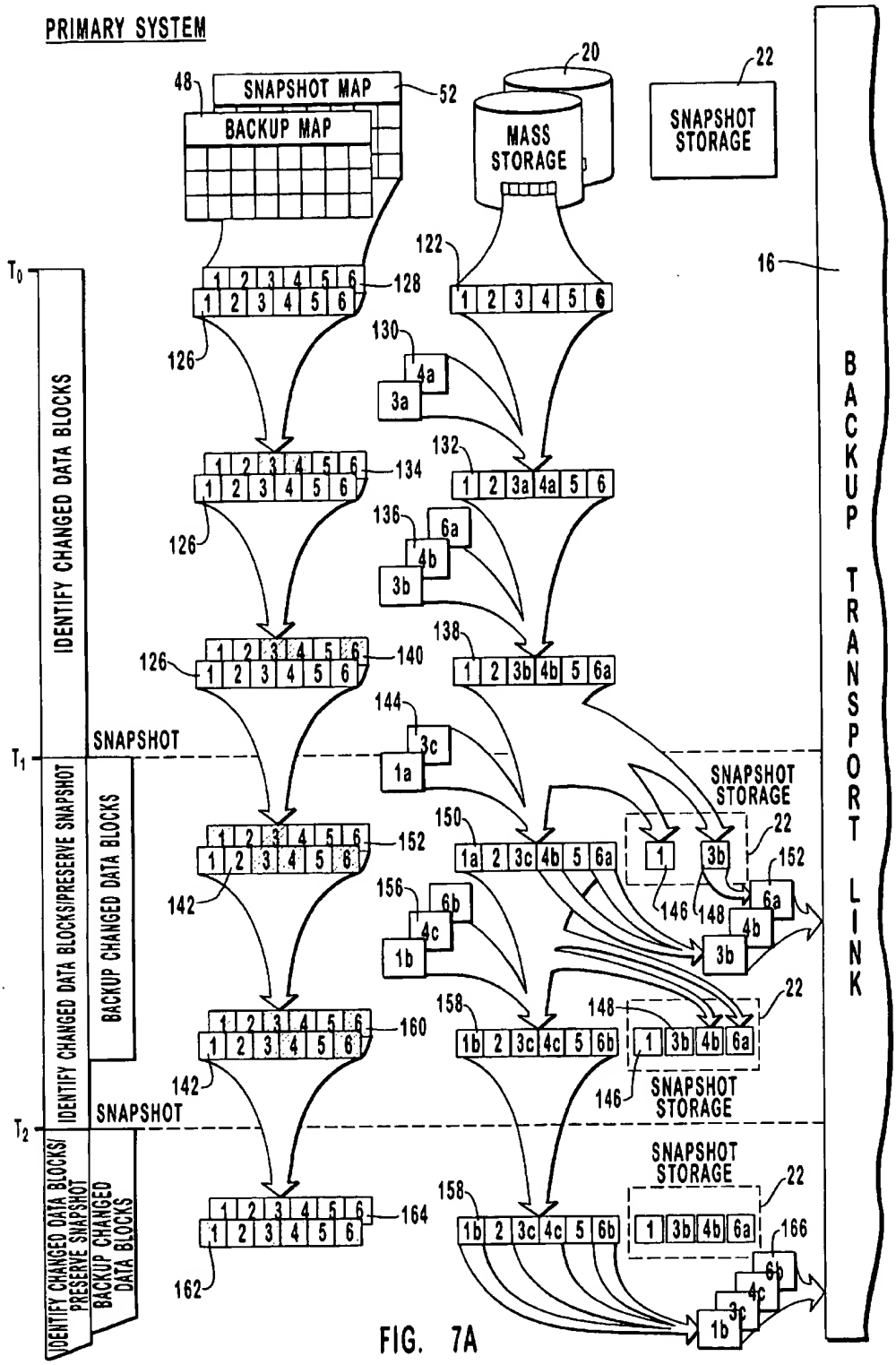

PRIMARY SYSTEM
SNAPSHOT MAP
BACKUP MAP
MASS STORAGE
SNAPSHOT STORAGE
BACKUP TRANSPORT LINK
IDENTIFY CHANGED DATA BLOCKS
IDENTIFY CHANGED DATA BLOCKS/PRESERVE SNAPSHOT
BACKUP CHANGED DATA BLOCKS
IDENTIFY CHANGED DATA BLOCKS/ PRESERVE SNAPSHOT
BACKUP CHANGED DATA BLOCKS
SNAPSHOT
SNAPSHOT
T0
T1
T2
48
52
20
22
16
122
128
126
130
132
134
136
138
140
144
150
152
142
156
158
160
162
164
146
148
166

FIG. 7A

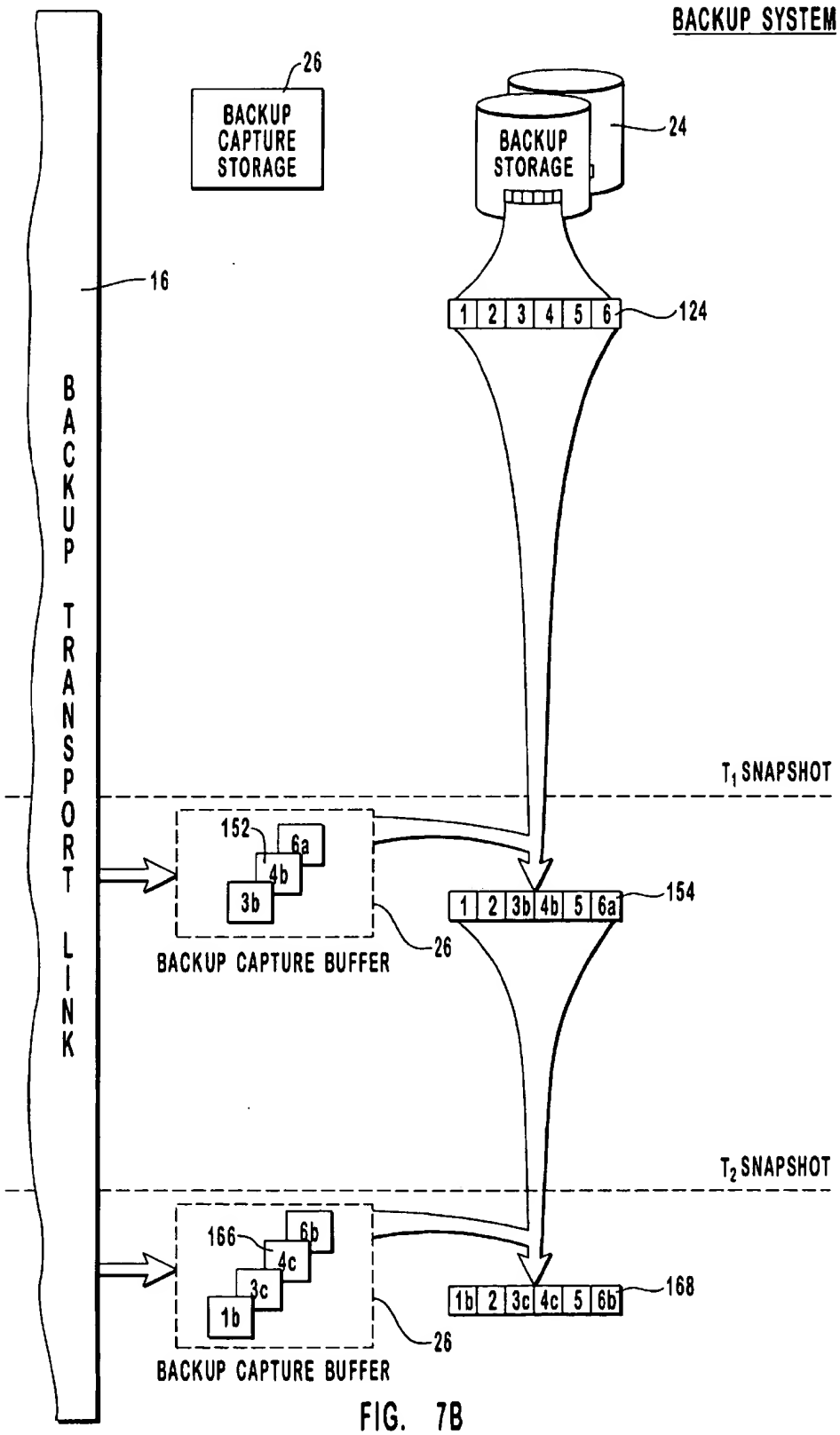

BACKUP SYSTEM
26
BACKUP CAPTURE STORAGE
BACKUP STORAGE
24
16
1 2 3 4 5 6
124
BACKUP TRANSPORT LINK
T1 SNAPSHOT
152
6a
4b
3b
BACKUP CAPTURE BUFFER
26
1 2 3b 4b 5 6a
154
T2 SNAPSHOT
166
6b
4c
3c
1b
BACKUP CAPTURE BUFFER
26
1b 2 3c 4c 5 6b
168

FIG. 7B

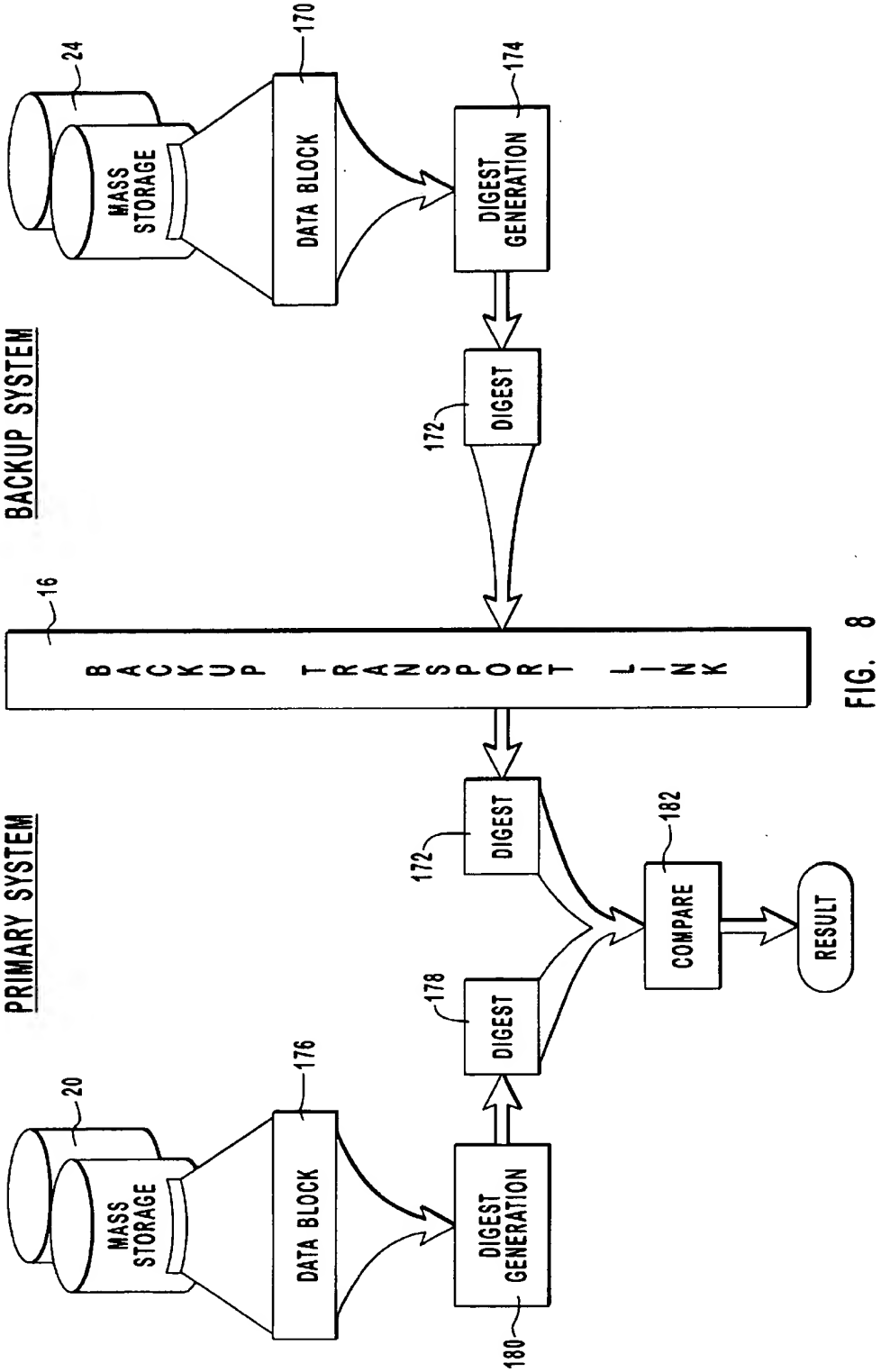

BACKUP SYSTEM
PRIMARY SYSTEM
MASS STORAGE
DATA BLOCK
DIGEST GENERATION
DIGEST
BACKUP TRANSPORT LINK
COMPARE
RESULT
24
170
174
172
16
182
178
20
176
180

FIG. 8

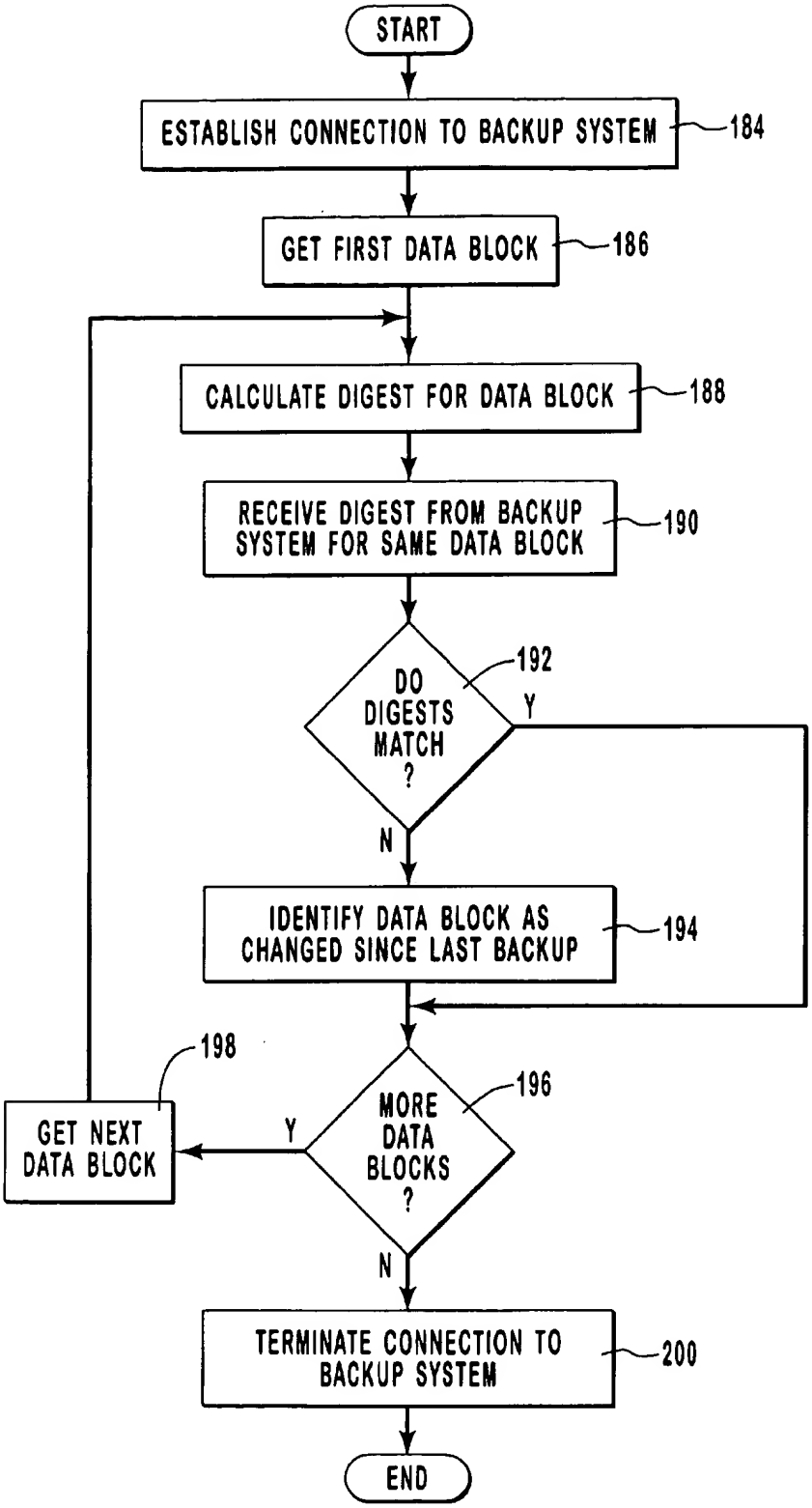

START
ESTABLISH CONNECTION TO BACKUP SYSTEM
184
GET FIRST DATA BLOCK
186
CALCULATE DIGEST FOR DATA BLOCK
188
RECEIVE DIGEST FROM BACKUP SYSTEM FOR SAME DATA BLOCK
190
DO DIGESTS MATCH ?
192
Y
N
IDENTIFY DATA BLOCK AS CHANGED SINCE LAST BACKUP
194
MORE DATA BLOCKS ?
196
Y
N
GET NEXT DATA BLOCK
198
TERMINATE CONNECTION TO BACKUP SYSTEM
200
END

FIG. 9

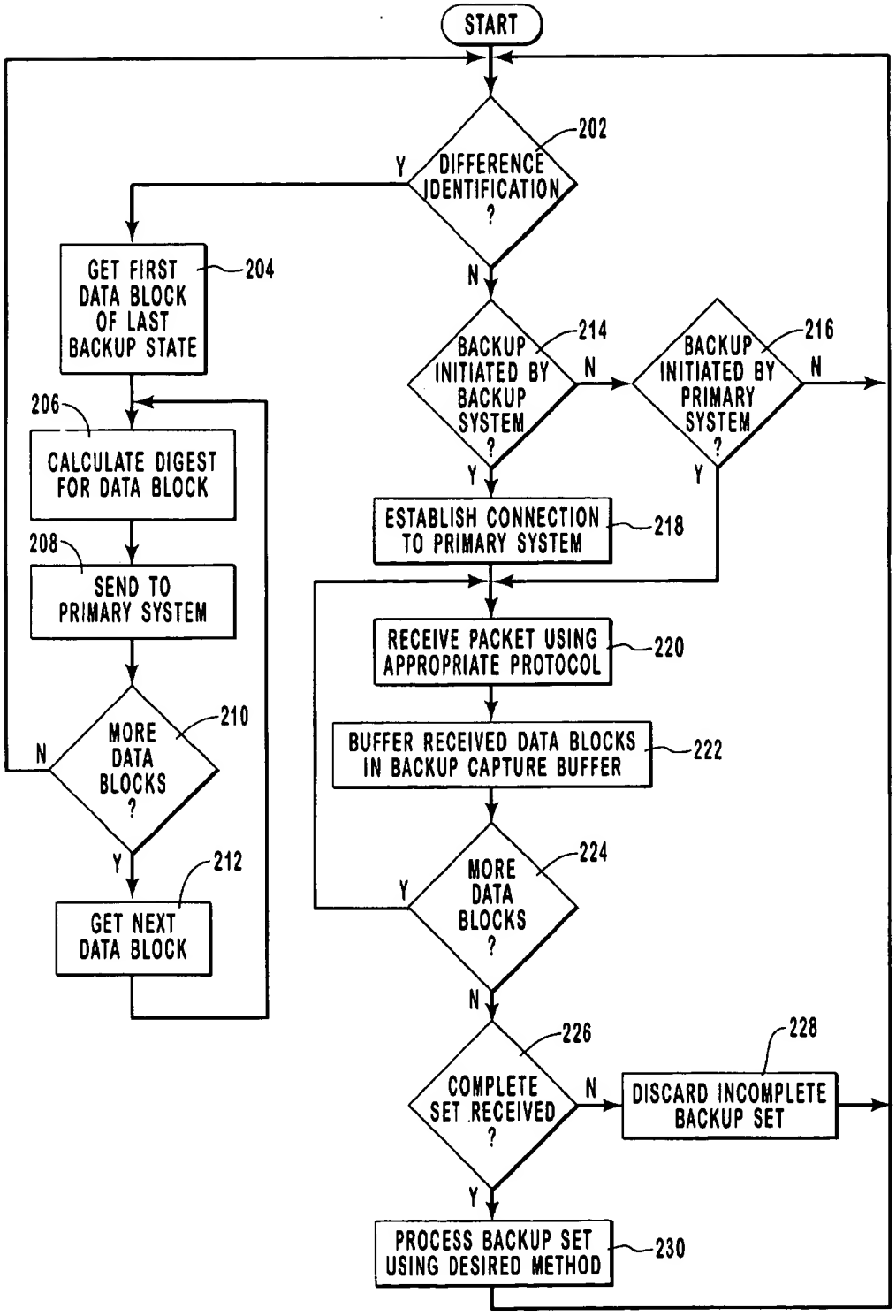

START
202
DIFFERENCE IDENTIFICATION ?
Y
N
GET FIRST DATA BLOCK OF LAST BACKUP STATE
204
206
CALCULATE DIGEST FOR DATA BLOCK
208
SEND TO PRIMARY SYSTEM
210
MORE DATA BLOCKS ?
N
Y
212
GET NEXT DATA BLOCK
214
BACKUP INITIATED BY BACKUP SYSTEM ?
N
Y
216
BACKUP INITIATED BY PRIMARY SYSTEM ?
N
Y
ESTABLISH CONNECTION TO PRIMARY SYSTEM
218
RECEIVE PACKET USING APPROPRIATE PROTOCOL
220
BUFFER RECEIVED DATA BLOCKS IN BACKUP CAPTURE BUFFER
222
224
MORE DATA BLOCKS ?
Y
N
226
COMPLETE SET RECEIVED ?
N
Y
228
DISCARD INCOMPLETE BACKUP SET
PROCESS BACKUP SET USING DESIRED METHOD
230

FIG. 10

COMPARING MASS STORAGE DEVICES THROUGH DIGESTS THAT ARE REPRESENTATIVE OF STORED DATA IN ORDER TO MINIMIZE DATA TRANSFER

RELATED APPLICATIONS

This application is a divisional of U.S. patent application Ser. No. 08/747,151 filed Nov. 8, 1996, and entitled BACKUP SYSTEM THAT TAKES A SNAPSHOT OF THE LOCATIONS IN A MASS STORAGE DEVICE THAT HAS BEEN IDENTIFIED FOR UPDATING PRIOR TO UPDATING, now U.S. Pat. No. 5,835,953, which is incorporated herein by reference. Which is a continuation-in-part of U.S. patent application Ser. No. 08/322,697 filed Oct. 13, 1994, now issued as U.S. Pat. No. 5,649,152 entitled METHOD AND SYSTEM FOR PROVIDING A STATIC SNAPSHOT OF DATA STORED ON A MASS STORAGE SYSTEM, which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The present invention relates to the protection of computer data, and more particularly to a system and method for backing up the data on one or more mass storage systems of one or more computers to a single backup system.

2. Present State of the Art

There is little question that computers have radically changed the way that businesses collect, manage, and utilize information. Computers have become an integral part of most business operations, and in some instances have become such an integral part of a business that when the computers cease to function, business operations cannot be conducted. Banks, insurance companies, brokerage firms, financial service providers, and a variety of other businesses rely on computer networks to store, manipulate, and display information that is constantly subject to change. The success or failure of an important transaction may turn on the availability of information which is both accurate and current. In certain cases, the credibility of the service provider, or its very existence, depends on the reliability of the information maintained on a computer network. Accordingly, businesses worldwide recognize the commercial value of their data and are seeking reliable, cost-effective ways to protect the information stored on their computer networks. In the United States, federal banking regulations also require that banks take steps to protect critical data.

Critical data may be threatened by natural disasters, by acts of terrorism, or by more mundane events such as computer hardware and/or software failures. Although these threats differ in many respects, they all tend to be limited in their geographic extent. Thus, many approaches to protecting data involve creating a copy of the data and placing that copy at a safe geographic distance from the original source of the data. Geographic separation may be an important part of data protection, but does not alone suffice to fully protect all data.

Often the process of creating a copy of the data is referred to as backing up the data or creating a backup copy of the data. When creating a backup copy of data stored on a computer or a computer network, several important factors must be considered. First, a backup copy of data must be logically consistent. A logically consistent backup copy contains no logical inconsistencies, such as data files that are corrupt or terminated improperly. Second, a backup copy of

data must be current enough to avoid data staleness. The time between backups, which largely determines the staleness of the backup copy, must be sufficiently short so the data on the backup is still useful should it be needed. For certain applications, such as networks that store financial transactions, backups a week old may be useless and much more frequent backups are needed. How frequent backup copies can be made is a function of many factors such as whether the backup can be made during normal business operations, the time it takes to make a backup copy, and so forth.

In order to create a backup copy of the data, several approaches have been taken. Each of the approaches has certain advantages and disadvantages. Perhaps the simplest approach to creating a backup copy of critical data is to copy the critical data from a mass storage system, such as the magnetic storage system utilized by a computer network, to a second archival mass storage device. The second archival mass storage device is often a storage device designed to store large amounts of data at the expense of immediate access to the data. One type of archival storage commonly utilized is magnetic tape. In these backup systems, data is copied from the mass storage system to one or more magnetic tapes. The magnetic tapes are then stored either locally or at a remote site in case problems arise with the main mass storage system. If problems arise with the mass main storage system, then data may be copied from the magnetic tape back to either the same or a different mass storage system.

Although utilizing magnetic tape or other archival storage as a means to guard against data loss has the advantage of being relatively simple and inexpensive, it also has severe limitations. One such limitation is related to how such backups are created. When data is copied from a mass storage system to a backup tape, the copy process generally copies the data one file at a time. In other words, a file is copied from the mass storage system onto the tape. After the copy is complete, another file is copied from the mass storage system to the tape. The process is repeated until all files have been copied.

In order to ensure the integrity of data being stored on the tape, care must be taken to keep the file from changing while the backup is being made. A simple example will illustrate this point. Suppose a file stores the account balances of all banking customers. If the account balances were allowed to change during the time the file is being backed up, it may be possible to leave a file in a logically inconsistent state. For example, if one account balance was backed up, and immediately after the account was backed up the account balance was debited $100.00, and if that same $100.00 was credited to a second account, then a situation may arise where the same $100.00 is credited to two different accounts.

In order to prevent such a situation from occurring, the data in a file must not change while the backup copy is made. A simple way to prevent data from changing is to prevent all access to the file during the backup procedure. In such a scheme, access to the files is cut off while the file is backed up. This approach is utilized by many networks where access to the mass storage system can be terminated after the close of business. For example, if a business closes at the end of each day and leaves its computer network essentially unused at night, user access to the network can be terminated at night and that time used to perform a backup operation. This, however, limits creation of a backup copy to once per day at off hours. This may be insufficient for some operations.

An increasing number of computer networks are used by computer businesses that operate world wide, and hence

these networks may be needed twenty-four hours a day, seven days a week. Shutting down such a network for several hours each day to make a tape backup may have a significant adverse affect on the business. For such businesses, creating a backup tape in the traditional manner is simply impractical and unworkable.

In an attempt to accommodate such operations or to increase the frequency of backups, an approach to copying data stored on computer networks known as "data shadowing" is sometimes used. A data shadowing program cycles through all the files in a computer network, or through a selected set of critical files and checks the time stamp of each file. If data has been written to the file since the last time the shadowing program checked the file's status, then a copy of the file is sent to a backup system. The backup system receives the data and stores it on tapes or other media. The shadow data is typically more current than data restored from a tape backup, because at least some information is stored during business hours. However, shadow data may nonetheless be outdated and incorrect. For example, it is not unusual to make a data shadowing program responsible for shadowing changes in any of several thousand files. Nor is it unusual for file activity to occur in bursts, with heavy activity in one or two files for a short time, followed by a burst of activity in several other files. Thus, a data shadowing program may spend much of its time checking the status of numerous inactive files while several other files undergo rapid changes. If the system crashes, or becomes otherwise unavailable before the data shadowing program gets around to checking the critical files, data may be lost.

Another problem with data shadowing programs is that they typically do not work for data kept in very large files. Consider a system with a single very large database and several much smaller data files. Assuming that a business's primary information is stored in the large database, it is reasonable to expect that a large percentage of the business day will be spent reading and writing data to the very large database. Assuming that a backup copy could be made of the very large database, the time needed to make a backup copy of such a large database may make the use of data shadowing impractical. The data shadowing program may attempt to make copy after copy of the large database. Making such numerous copies not only takes a tremendous amount of time, but also requires a tremendous amount of backup storage space.

Another problem of data shadowing type systems is that open files are generally not copied. As previously described, a file must be frozen while a backup copy is made in order to prevent changes to the file during the backup process. Thus, data shadowing systems usually do not attempt to make copies of open files. If changes are constantly being made to large database, the large database will constantly be open and data shadowing systems may not copy the database simply because the file is open. For at least these reasons, data shadowing systems are typically not recommended for very large data files.

Another approach that has been attempted in order to overcome some of these limitations is a process whereby a time sequence of data is captured and saved. For example, many systems incorporate disk mirroring or duplexing. In disk mirroring or duplexing, changes made to a primary mass storage system are sent to another backup or secondary mass storage systems. In other words, when a data block is written to the primary mass storage system, the same data block is written to a separate backup mass storage system. By copying each write operation to a second mass storage system, two mass storage systems may be kept synchronized so that they are virtually identical at the same instant in time. Such a scheme protects against certain types of failures, but remains vulnerable to other types of failures.

The primary type of failure that disk mirroring overcomes is a hardware failure. For example, if data is written to two disks simultaneously, then if one disk fails, the data is still available on the other disk. If the two disks are connected to two separate disk controller cards, then if a single disk controller card or a single disk fails, then the data is still accessible through the other disk controller card and disk assembly. Such a concept can be extended to include entire systems where a secondary network server mirrors a primary server so that if a failure occurs in the primary network server, the secondary network server can take over and continue operation. The Novell® SFT line of products utilize variants of this technology.

While such systems provide high reliability against hardware failures and also provide almost instantaneous access to backup copies of critical data, they do not guard against software failures. As software becomes more and more complex the likelihood of software failures increase. In today's complex computing environments where multiple computer systems running multiple operating systems are connected together in a network environment, the likelihood of software errors causing occasional system crashes increases. When such a software error occurs, both the primary mass storage system and the mirrored mass storage system may be left in a logically inconsistent state. For example, suppose that a software error occurred during a database update. In such a situation, both the primary mass storage system and the mirrored mass storage system would have received the same write command. If the software error occurred while issuing the write command, both mass storage systems may be left in an identical, logically inconsistent state. If the mirrored mass storage system was the only form of backup in the network, critical data could be permanently lost.

If the backup is to be made at a remote location, the problems with the above technology are exacerbated. For example, if disk mirroring is to be made to a remote site, the amount of data transferred to the remote site can be considerable. Thus, a high speed communication link must exist between the primary site and the secondary or backup site. High speed communication links are typically expensive. Furthermore, if a time sequence of data is to be sent to a backup system at a remote location over a communication link, then the reliability of the communication link becomes a significant issue. If for any reason the communication link should be temporarily severed, synchronization between the primary mass storage system and the secondary or backup mass storage system would be lost. Steps must then be taken to reconcile the two mass storage devices once the communication link is reestablished. Thus, mirroring a primary mass storage system at a remote site is typically difficult and very expensive.

The problems of backing up a single system to a remote site becomes even more complicated when a single remote site is to service several primary systems. Using a file-by-file backup method requires a significant amount of time if the mass storage devices of the primary systems are relatively large. In such a situation, a single night may not be sufficient to backup all primary sites to a single remote site. Thus, in some situations, a file-by-file transfer method cannot be used. Similar problems exist with remote disk mirroring technology. Since a remote disk mirror typically requires a dedicated communication link, the backup system must be

sufficiently fast to handle communications from a plurality of dedicated communication lines. The amount of data that must be received and stored by the backup system may quickly overwhelm the capabilities of the backup system.

It would, therefore, represent an advancement in the art to have a backup system that could ensure that a logically consistent backup was created. It would also represent an advancement in the art to have a backup system that allowed data to be backed up without terminating user access to the mass storage system. Furthermore, it would be highly desirable to have a backup system that could backup open computer files. It would also be extremely desirable to have a backup system that could backup files as they changed in order to reduce the time needed to create a backup copy. It would be highly desirable to have a single backup system that could service a plurality of primary systems. Finally, it would also represent an advancement in the art to have a backup system that could accomplish these functions either locally or remotely using a low bandwidth communication link.

SUMMARY AND OBJECTS OF THE INVENTION

The foregoing problems in the prior state of the art have been successfully overcome by the present invention, which is directed to a system and method for backing up a primary mass storage device to a backup storage device. The current system and method provides three significant advantages over the prior art. First, the backup system and method of the present invention reduces the amount of data needed to make a backup by backing up only those data blocks of the primary mass storage device that change. Second, the system and method of the present invention emphasize security of the backup by ensuring that the primary storage device is in a logically consistent state when a backup is made. Third, because the data needed to make a backup is reduced to the absolute minimum, and because backups are only made of logically consistent states, backup frequency can be increased in order to capture many more logically consistent states. The backup system and method of the present invention accomplishes this without terminating user access to the mass storage system.

The present invention begins with the assumption that a primary mass storage device connected to a primary computer and a backup storage system connected to a backup storage device contain identical data. This may be accomplished, for example, by making a complete copy of the primary mass storage device to the backup device using either traditional backup techniques or traditional disk mirroring techniques. Once the primary mass storage device and the backup storage device contain the same data, the present invention tracks the changes made to the primary mass storage device. This tracking is done by identifying those storage locations on the primary mass storage device that have new data written in them from the time that the backup storage device was in sync with the primary mass storage device. By identifying those changes that have been made to the primary mass storage device, the invention identifies those changes that need to be made to the backup storage device in order to bring the backup storage device current with the primary mass storage device.

Once the changes that need to be made to the backup storage device have been identified, the changes are sent to the backup system. The backup system then has available all data to bring the backup storage device current with the primary mass storage device. In order to preserve the original data of the primary mass storage device during the backup process, a static snapshot of the primary mass storage device is taken. This static snapshot captures the changes that have been made to the primary mass storage device and that need to be transferred to the backup system. In order to make the backup transparent to users, it is preferred that the static snapshot be taken in such a way that user access to the primary mass storage device is not interrupted.

The present invention includes a mechanism to identify when the primary mass storage device is in a logically consistent state in order to determine when a static snapshot should be made. By identifying a logically consistent state and then taking a static snapshot of the chances made up to that point in time, when the changes are transferred to the backup system, the backup system is guaranteed to capture a logically consistent state. By capturing snapshots of succeeding logically consistent states, the backup can capture one logically consistent state after another. In this way, if the backup data should ever be needed, the backup data will be in a logically consistent state. The backup system moves from one logically consistent state to another logically consistent state thus eliminating one of the problems of the prior art.

Because the present invention takes a data block approach to the backing up of a mass storage system, the present invention minimizes the amount of data that needs to be transferred to make a backup to the absolutely minimum possible. For example, if a large database has five records that change, prior art systems would copy the entire large database. The present invention, however, copies only the five records that have changed. Because the amount of data is minimized, the present invention is particularly well suited to backing up data to a backup system located at a remote site. The present invention can utilize low bandwidth communication links to transfer backup data to a remote backup site. As an example, in many cases conventional dial-up telephone lines with a 56.6 k baud modem will be entirely adequate for many situations.

Because the data needed to make a backup copy is minimized through the present invention, a series of backup copies may be made, one after the other. This allows the state of a single mass storage system to be captured with greater frequency. In addition, a single centralized backup system may support a plurality of primary servers so that each can be backed up to the same backup system.

Accordingly, it is a primary object of the present invention to provide a system and method for mass storage backup that minimizes the amount of data that needs to be transferred to a backup system.

Another central object of the present invention is to provide a system and method for mass storage backup that can capture logically consistent states so that the backup is not left in a logically inconsistent state.

Yet another object of the present invention is to allow the backup system to capture successive logically consistent backup states in order to provide a series of logically consistent backup states.

Additional objects and advantages of the present invention will be set forth in the description which follows, and in part will be obvious from the description, or it may be learned by practice of the invention. The objects and advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the attended claims. These and other objects and features of the present invention will become more fully

apparent from the following description and appending claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

FIG. **1** is block diagram representing a system of the present invention;

FIG. **2** is a diagram illustrating the timing of one method of the present invention;

FIG. **3** is a system level block diagram of one embodiment of the present invention;

FIG. **4** illustrates the processing details of one embodiment of the mass storage read/write processing block of FIG. **3**;

FIG. **5** illustrates the processing details of one embodiment of the primary backup processing block of FIG. **3**;

FIG. **6** illustrates the processing details of one embodiment of the backup read processing block of FIG. **3**;

FIGS. **7**A and **7**B are diagrams illustrating an example of a method of the present invention;

FIG. **8** illustrates a method of identifying differences between a mass storage system and backup storage system;

FIG. **9** illustrates the processing details of one embodiment of the difference identification processing block of FIG. **3**.

FIG. **10** illustrates the processing details of one embodiment of backup system processing block of FIG. **3**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The following invention is described by using flow diagrams to describe either the structure or the processing of certain embodiments to implement the system and method of the present invention. Using the diagrams in this manner to present the invention should not be construed as limiting of its scope. The present invention contemplates both a system and method for backing up a primary mass storage device to a backup storage device. The presently preferred embodiment of the system for backing up a primary mass storage device to a backup storage device comprises one or more general purpose computers. The system and method of the present invention, however, can also be used with any special purpose computers or other hardware systems and all should be included within its scope.

Embodiments within the scope of the present invention also include computer-readable media having encoded therein computer-executable instructions. Such computer-readable media can be any available media which can be accessed by a general purpose or special purpose computer. By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, magneto-optical storage devices, or any other medium which can be used to store the desired program code means and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media. In turn, registers of a CPU or other processing unit that store computer-executable instructions while decoding and executing the same are also included within the scope of the computer-readable media.

Computer-executable instructions comprise, for example, executable instructions and data which cause a general purpose computer or special purpose computer to perform a certain function or a group of functions.

Referring now to FIG. **1**, a system level block diagram of one embodiment of the present invention is presented. The system, shown generally as **10**, comprises one or more primary systems **12**, a backup system **14**, and backup transport means for transporting data between primary system **12** and backup system **14**. In FIG. **1**, the backup transport means is illustrated as backup transport link **16**. In FIG. **1**, primary system **12** may be any type of networked or stand alone computer system. For example, primary system **12** may be a network server computer connected to a computer network such as computer network **18**. Primary system **12** may also be a stand alone system. Primary system **12** may also be a backup or standby server of a computer network connected to a primary server. The present invention can be used with any type of computer system. In this sense, the term "primary" is not meant to define or describe a computer system as a primary network server (as opposed to a backup or standby network server). In this description, the term "primary" is used to refer to the fact that the system has attached mass storage means for storing a copy of the data that is to be backed up. In other words, the term "primary" is used to differentiate the system from backup system **14**.

Primary system **12** has attached thereto mass storage means for storing a plurality of data blocks in a plurality of storage locations. Each of the storage locations is specified by a unique address or other mechanism. Mass storage means can be any storage mechanism that stores data which is to be backed up using the present invention. For example, such mass storage means may comprise one or more magnetic or magneto-optical disk drives. It is, however, presumed that such mass storage means has a plurality of storage locations that can be used to store data blocks. The storage locations are addressed by a unique address or index so that a particular data block may be written thereto or retrieved therefrom. In FIG. 1, for example, such mass storage means is illustrated by mass storage device **20**.

The term "data block" will be used to describe a block of data that is written to or read from mass storage means. The term "data block" is intended to be broadly construed and should include any size or format of data. For example, the data stored in an individual sector on a disk is properly referred to as a data block. The amount of data stored in a group or cluster of sectors may also properly be referred to as a data block. If the mass storage means is a RAM or other word or byte addressable storage device, the term data block may be applied to a byte, a word, or multiple word unit of data.

As described in greater detail below, embodiments within the scope of this invention use a static snapshot of all or part of the mass storage device during the backup process. Embodiments within the scope of this invention therefore comprise preservation memory means for storing data

blocks of said mass storage means so as to create a static snapshot of the mass storage means at a particular point in time. As described in greater detail below, such preservation memory means may comprise any type of writable storage device such as RAM, EEPROM, magnetic disk storage, and the like. Such preservation memory means may also comprise a portion of mass storage device **20**. In FIG. **1**, such preservation memory means is illustrated, for example, by snapshot storage device **22**. Preservation memory means is discussed in greater detail below.

Since primary system **12** may be any type of general purpose or special purpose computer, primary system **12** may also comprise any other hardware that makes up a general purpose or special purpose computer. For example, primary system **12** may also comprise processor means for executing programmable code means. The processor means may be a microprocessor or other CPU device. The processor means may also comprise various special purpose processors such as digital signal processors and the like. Primary system **12** may also comprise other traditional computer components such as display means for displaying output to a user, input means for inputting data to primary system **12**, output means for outputting hard copy printouts, memory means such as RAM, ROM, EEPROM, and the like.

Backup system **14** of FIG. **1** comprises backup storage means for storing data blocks received from primary system **12**. Backup storage means can comprise any type of storage device capable of storing blocks of data received from a primary system. For example, backup storage means may comprise a storage device identical to the mass storage device of a primary system. If the primary system has a large magnetic disk, for example, the backup storage means may also comprise a large magnetic disk. If the backup storage means is the same as the mass storage means of the primary system, the backup storage means can closely mirror the mass storage means of the primary system. As another example, backup storage means may comprise archival storage devices such as a magnetic tape drive or an optical or magneto-optical drive. The type of storage devices that may be used for backup storage means is limited only by the particular application where they are utilized. In some situations it may be more desirable to have a backup storage means that more closely resembles the mass storage means of the primary system. In other situations it may be perfectly acceptable to have archival type storage means that are optimized to store large amounts of data at the expense of rapid access. All that is required is that the backup storage means be able to store data blocks transferred to the backup system from the mass storage means of the primary system. In FIG. **1** the backup storage means is illustrated by backup storage device **24**.

As described in greater detail below, backup storage system **14** may comprise backup capture means for storing data blocks transferred to backup system **14** until all such data blocks have been received. Because the present invention transfers only certain data blocks, a situation can arise where a logically inconsistent state is created if only some of the data blocks are applied to backup storage device **24**. In order to prevent this from happening, it may be desirable to save the transferred data blocks in a separate location, such as the backup capture means, until all data blocks have been received. This ensures that a complete group of data blocks are received before any action is taken. Backup capture means can comprise any type of storage that can store data blocks received from primary system **12**. For example, backup capture means may comprise RAM, magnetic disk storage, or any other storage medium. It is preferred, however, that backup capture means have sufficient speed to be able to store data blocks as they are received. Backup capture means must also provide data blocks to backup system **14** so that backup system **14** can transfer the data blocks to its attached backup storage means. In FIG. **1**, the backup capture means is illustrated by backup capture buffer **26**.

In order to transfer data between primary system **12** and backup system **14**, backup transport link **16** is used. Backup transport link **16** is one illustration of backup transport means for transporting data between primary system **12** and backup system **14**. Backup transport link **16** may comprise any combination of hardware and/or software needed to allow data communications between primary system **12** and backup system **14**. For example, backup transport link **16** may be a local area network (LAN), a wide area network (WAN), a dial-up connection using standard telephone lines or high speed communication lines, the internet, or any other mechanism that allows data to flow between primary system **12** and backup system **14**. As explained in greater detail below, the present invention is designed to minimize the amount of data that flows between primary system **12** and backup system **14** so that only that data necessary to bring backup storage means, such as backup storage device **12**, current with respect to the primary mass storage means, such as mass storage device **20** is transferred. This allows backup transport link **16** to encompass a wider variety of technologies that cannot be used with prior art systems. The bandwidth requirements for backup transport link **16** are typically very modest and a 56.6 k baud dial-up connection will be entirely adequate for many purposes.

Referring next to FIG. **2**, an overview of the method used to backup a mass storage means, such as mass storage device **20** of FIG. **1**, to a backup storage means, such as backup storage device **24** of FIG. **1**, is presented. Initially, the method illustrated in FIG. **2** presumes that the mass storage device and the backup storage device are current. In other words, the backup storage device contains a copy of the data stored on the mass storage device. This may be accomplished using any number of conventional technologies. The type of technology used will depend in large measure on the type of media used for the backup storage device. For example, if the backup storage device is a disk similar to a disk used for the mass storage device, then disk mirroring or other means may be used to copy the data from the mass storage device to the backup storage device. On the other hand, if the backup storage device utilizes magnetic tape or other archival type storage, then a backup may be made in the conventional way that such archival tape backups are made. In FIG. **2**, the backup storage device is assumed to have a current copy of the data stored on the mass storage device at time T_0.

Beginning at time T_0, the method summarized in FIG. **2** maintains the backup storage device in a current state with respect to the mass storage device. The method summarized in FIG. **2** captures successive logically consistent states. This results in the backup storage device either moving from one logically consistent state to a subsequent logically consistent state or allows the backup storage device to capture succeeding logically consistent states. This creates a tremendous advantage over prior art systems which may leave the backup storage device in a logically inconsistent state. By ensuring that the backup device is in a logically consistent state, the present invention ensures that a useable backup is always available.

Returning now to FIG. **2**, beginning at time T_0 the changes to the mass storage system are tracked. This is

illustrated in FIG. 2 by block **28**. The changes are preferably tracked by identifying storage locations of the mass storage device that have new data written in them starting at time T_0. As explained in greater detail below, this may be done by keeping a map which identifies those storage locations that have new data written in them starting with time T_0. Alternatively, a list of the storage locations that have new data written in them beginning at time T_0 may be kept.

At some point in time, it is desirable to capture the changes that have been made and to transfer those changes to the backup system. In a preferred embodiment, the system identifies logically consistent state of the mass storage device and takes a static snapshot of at least those storage locations that have been changed since time T_0. In FIG. **2**, the logically consistent state is identified as time T_1 and a snapshot is taken.

A static snapshot is designed to preserve data as it is exists at a particular point in time so that the data will be available after that particular point in time in its original state even though changes are made to the mass storage system after the snapshot time. Many ways exist of creating such a static snapshot. Any method will work with the present invention, however, some methods are preferred over others due to various advantages. The details of how a static snapshot is taken and a preferred method for creating a static snapshot is presented below. For this summary, however, it is important to understand that any method which creates a static snapshot can be used with the present invention. It is, however, preferred that the static snapshot be taken without terminating user read or write access to the mass storage device.

At time T_1, the changes identified between time T_0 and time T_1 are backed up by sending them to the backup storage device. This is illustrated in FIG. **2** by arrow **30** and block **32**. The changes are sent to the backup storage device by sending the data stored in only those storage locations where new data was written between time T_0 and time T_1. Since the data is preserved by a snapshot at time T_1, the data will be available for transfer to the backup storage device even though new data is written to the mass storage device after time T_1. The map or other mechanism that was used to track which storage locations had data written therein between time T_0 and time T_1 is used to identify the data that should be transferred to the backup storage device. Note that only those data blocks that were changed between time T_0 and T_1, are transferred. Thus, only incremental changes are sent and entire files are not transferred unless the entire file changes.

As explained in greater detail below, as the data is received by the backup storage device, it is preferably buffered in a temporary location until all the data from time T_0 to time T_1 has been transferred. Once all the data has been transferred, then the changes may be applied to the backup storage device in order to bring the backup storage device current to time T_1. Alternatively, the changes between time T_0 and T_1 may be kept as an incremental backup so that the logically consistent state at time T_0 and the logically consistent state at time T_1 can be reconstructed if desired.

Since new data may be written to the mass storage device after time T_1 while the backup is being performed, a mechanism must be in place to identify the changes that are made after time T_1 if another backup is to be made after time T_1. In FIG. **2**, the changes after time T_1 are tracked as indicated by block **34**. This will allow the changes made after time T_1 to also be transferred to the backup storage device in order to bring the backup storage device current to some later time.

As illustrated in FIG. **2**, the sequence described above repeats itself at time T_2. This is illustrated by arrow **36**, block **38**, and block **40**. As described previously, the snapshot taken at time T_2 should represent logically consistent state so that when the changes made between times T_1 and T_2 are transferred to the backup storage device, the backup storage device is brought current to the logically consistent state at time T_2.

From the summary given above, several observations can be made. The first observation is that the present invention backs up only the data stored in the storage locations that were changed since the last backup. This creates a significant advantage over the prior art. For example, consider a database where only a very few data records are changed. Prior art systems would attempt to backup the entire database if a change had been made. The present invention, however, only backs up those data blocks that have been changed due to the few records that were changed. This means that the time needed to make the backup of the database and the storage requirements to make the backup of the database are dramatically reduced over the prior art.

Another important difference from the prior art is highlighted in the above description. The present invention captures the data as it is exists when the snapshot is taken. The present invention does not try to send to the backup storage device the time sequence of changes that were made to the mass storage device. For example, if a single record of the database was changed ten times between the time the last backup was made and the current backup time, certain prior art systems would send ten changes to the backup storage device. The present invention, however, simply sends the last change that was made before the current backup time. In this example, such a scheme reduces the amount of data sent to the backup device by ten times. The present invention reduces the amount of data sent to the backup device to the very minimum needed to make a logically consistent backup. This allows the communication link between the primary system and the backup system to be much lower bandwidth than prior art systems. The present invention is, therefore, ideally suited to embodiments where the backup system is situated at a remote site from the primary system. When the backup system is situated at a remote site, conventional dial-up telephone lines may be used to transfer backup data between the primary system and the backup system.

The present invention also supports a many-to-one backup embodiment. For example, consider the situation presented in FIG. I where an embodiment comprises a single backup system and a plurality of primary system. The backup system could be situated either remotely or locally. The backup system could then initiate contact with one primary system, receive the changes that have occurred since the last backup of that system, and terminate the connection. A connection would then be established to another primary system and the backup system could receive the changes that occurred on that primary system since the last backup. Thus, the backup system contacts each primary system in turn and receives the changes that have occurred since the last time the primary system was contacted. Such an embodiment may be of great value to a business with many branch offices where copies of the data from these branch offices are to be stored at a central location.

Turning next to FIG. **3**, a top level diagram of one embodiment to implement the method summarized in FIG. **2** is presented. The following description presents a top level overview of each of the processing blocks illustrated in FIG. **3**. The details of each processing block are then presented.

During normal operation of a computer system, data is periodically written to or read from attached mass storage means such as mass storage device **20**. Embodiments within the scope of this invention therefore comprise means for writing data to a mass storage device and means for reading data from a mass storage device. In FIG. **1**, such means is illustrated, for example, by mass storage read/write processing block **42**. Although the details of mass storage read/write processing block **42** are presented later, the basic function of mass storage read/write processing block **42** is to write a data block to an identified storage location on mass storage device **20** or read a data block from an identified storage location on mass storage device **20**. In FIG. **3**, requests to read or write a data block from or to an identified storage location are illustrated by mass storage read/write requests **44**. Whenever a read or write is requested, mass storage read/write processing block **42** can return a response as illustrated by mass storage read/write response **46**. The responses can include a completion code or other indicator of the success or failure of the requested operation and, in the case of a read request, the data requested.

As described in conjunction with FIG. **2**, a method of the present invention tracks changes that occur between a first instant in time and a second instant in time. Embodiments within the scope of this invention therefore comprise means for identifying which storage locations of mass storage device **20** have had new data stored therein between a first instant in time and a second instant in time. Any method for identifying and tracking such locations can be utilized with the present invention. All that is necessary is that the storage locations that have had new data stored in them since the last backup be able to be identified. In FIG. **3** such means is illustrated, for example, by backup map **48**. Backup map **48** may comprise a boolean entry for each storage location on mass storage device **20**. When a storage location has new data written in it, the entry for the storage location may then be set. Alternatively, a list of storage locations that have new data stored in them may also be kept. All that is required is the ability to distinguish and identify storage locations that have had new data stored therein since a particular point in time.

As previously described, when a backup is to be made, a static snapshot of at least the storage locations that are to be backed up is made. Embodiments within the scope of this invention therefore comprise means for preserving a static snapshot at a particular instant in time. The use of a static snapshot to preserve at least the storage locations that are to be backed up to a backup system is preferred because it allows users to continue to access mass storage device **20** while the changes are being backed up. Since it takes a period of time to transfer the changes from the primary system to the backup system, the data that is to be transferred must remain unchanged until it is transferred. One way to ensure that the data remains unchanged is to prevent access to mass storage device **20**. This will prevent any data from being written to mass storage device **20** and ensures that the data to be backed up remains unchanged until it can be transferred to the backup system. Unfortunately, this solution is highly undesirable. It is, therefore, preferred that when changes are to be transferred to the backup system, a static snapshot of at least the data that will be transferred is taken. Such a static snapshot will preserve the data to be transferred in its original condition until it can be transferred while simultaneously allowing continued access to mass storage device **20** so that data can continue to be written thereto or read therefrom.

Any method of preserving a static snapshot can be used with the present invention. However, it is preferred that whatever method is used be able to preserve a static snapshot without interrupting access to mass storage device **20**. In other words, it is preferred that the static snapshot be preserved in such a way that users can continue to read data from or write data to mass storage device **20**.

In FIG. **3**, the means for preserving a static snapshot is illustrated by snapshot processing block **50**. As illustrated in FIG. **3**, it may make sense to incorporate the snapshot processing mechanism into the mass storage read/write processing block. Although the details of snapshot processing block **50** are presented below, one preferred embodiment preserves a static snapshot by copying a data block that is to be overwritten from mass storage device **20** into snapshot storage **22** and then indicating that the block has been preserved in snapshot map **52**. Once a copy has been placed into snapshot storage **22**, then the copy of the data block on mass storage device **20** can be overwritten.

As described above in conjunction with FIG. **2**, if a series of successive backups are to be made, it is necessary to track the changes made to a mass storage device, such as mass storage device **20**, during the time that a backup is being made. In other words, it may be necessary to track changes made to mass storage device **20** after a snapshot is made. Embodiments within the scope of the present invention can comprise means for identifying the storage locations of a mass storage device that have new data stored therein after the point in time that a snapshot is made. Any type of mechanism that tracks and identifies storage locations of a mass storage device that have new data stored therein after a particular point in time can be utilized. For example, a map similar to backup map **48** may be used. As another example, a list of data locations that have new data stored therein after a particular point in time may also be used. Depending on the type of snapshot mechanism used, the snapshot mechanism may inherently track such information. In such an embodiment, this information may be saved for later use. In FIG. **3**, such means is illustrated by snapshot map **52**. As described in greater detail below, one implementation of a snapshot mechanism tracks storage locations with new data stored therein after the snapshot is made in a snapshot map, such as snapshot map **52** of FIG. **3**.

Embodiments within the scope of this invention comprise means for transferring data blocks that are to be backed up to a backup system. In FIG. **3** such means is illustrated, for example, by primary backup processing block **54**. Although the details of primary backup processing block **54** are presented in greater detail below, the general purpose of primary backup processing block **54** is to take data blocks that are to be backed up and transfer those data blocks to a backup system using an appropriate protocol. As described in conjunction with FIG. **2**, and as described in greater detail below, the data blocks to be transferred will be those data blocks that have been stored in storage locations on the mass storage device since the last backup.

Primary backup processing block **54** may incorporate functionality to either initiate a backup and transfer data to the backup system or respond to a backup initiated by the backup system. In this way, either the primary system or the backup system can initiate a backup. The details of how backups may be initiated by either the primary system or the backup system are presented in greater detail below.

In the discussion of FIG. **2** that presented an overview of a method of the present invention, a static snapshot was used to preserve the state of changed data blocks at a particular point in time. Those changed data blocks were then backed up to a backup system. If changed data blocks are preserved

by a static snapshot, then before the data blocks can be transferred to a backup system they must be retrieved from the snapshot. Embodiments within the scope of this invention may, therefore, comprise means for retrieving data blocks that were preserved by a static snapshot. Such means may be part of the means for transferring data blocks to the backup system or such means may be separate. In FIG. **3**, the means for retrieving data blocks that were preserved by a static snapshot is illustrated by backup read processing block **56**. The details of one embodiment of backup read processing block **56** are presented below. This processing block retrieves preserved data from its storage location and passes a retrieved data block to primary backup processing block **54** for transfer to the backup system. This functionality may also be incorporated into primary backup processing block **54**. However, in order to emphasis the function performed by backup read processing block **56**, the block is illustrated separately in FIG. **3**.

The present invention is designed to capture one or more logically consistent backup states at the backup system. In order to capture these logically consistent backup states, embodiments within the scope of this invention may comprise means for determining when a logically consistent state has been achieved. A logically consistent state is a state where no logical inconsistencies such as improperly terminated files exist on the mass storage system. A logically consistent state may be identified by a number of mechanisms. For example, a logically consistent state may be identified by watching the activity on the mass storage device. When no activity exists on a mass storage device, it may generally be presumed that all internal data buffers have been flushed and their data written to the mass storage system and the mass storage system is not in a state where data blocks are being updated. In addition, APIs may exist that can be called to identify when a logically consistent state has been reached. For example, the operating system or other program may have an API call that may be made that will return when a logically consistent state has been reached. As yet another example, the system may broadcast a message to all users connected to a network that a snapshot will be taken at a given time. Users can then take appropriate steps, if necessary, to ensure a logically consistent state of their files. Other mechanisms may also be used. As described in greater detail below, the means for determining when a logically consistent state has been achieved may be incorporated into one of the processing blocks of FIG. **3**, as for example, primary backup processing block **54**.

When successive backups are to be made to a backup system by the present invention, embodiments within the scope of this invention may comprise a mechanism or means for identifying differences that exist between the mass storage device of the primary system, as for example mass storage device **20** of FIG. **3**, and the backup storage device, as for example backup storage device **24** of FIG. **3**. Such a mechanism may be useful when, for whatever reason, it is unclear if differences exist between mass storage device **20** and backup storage device **24**. For example, suppose that the backup system crashed or the primary system crashed or otherwise became unavailable for a period of time. When the backup system or primary system again becomes available, it may be impossible to identify exactly what specific differences exist between mass storage device **20** and backup storage device **24**. It may, therefore, be desirable to identify any differences so that the data blocks which are different may be transferred from mass storage device **20** to backup storage device **24** in order to bring backup storage device **24** current with mass storage device **20**. Embodiments within the scope of this invention may therefore comprise means for identifying differences between data stored in the plurality of storage locations on a mass storage device and data stored on a backup storage device. In FIG. **3**, such means is illustrated, for example, by difference identification processing block **58**. Although the details of difference identification processing block **58** are presented below, the block is responsible to identify any differences that exist between mass storage device **20** and backup storage device **24**. This block can place appropriate entries into backup map **48** or snapshot map **52** to track identified differences.

Embodiments within the scope of this invention comprise a backup system that stores data blocks transferred from one or more primary systems. In FIG. **3**, the processing that occurs on such a backup system is illustrated by backup system processing block **60**. As discussed in greater detail below, backup system processing block **60** receives data blocks via backup transport link **16** and stores them on backup storage device **24**. As previously described, backup storage device **24** may be any type of storage device that can store the data blocks received from one or more primary systems. For example, the storage device may be a disk drive similar to a disk drive used for mass storage on a primary system. As another example, backup storage device **24** may be any archival storage medium such as magnetic tape. As another example, backup storage device **24** may be optical disks or a plurality of optical disks. All that is required is that the backup storage device be able to store the data blocks received from one or more primary systems in a format where they can be retrieved if necessary in order to recover data lost by a mass storage device at a primary system. If more than one primary system is serviced by a single backup system, it may be desirable to have separate backup storage devices for each primary system or it may be desirable to have a single backup storage device that serves all primary systems.

As illustrated in FIG. **3**, data packets are exchanged between backup system processing block **60** and one or more processing blocks on a primary system, such as primary backup processing block **54** and difference identification processing block **58**. These data packets are exchanged using a protocol appropriate to the amount of data transferred and the particular details of backup transport link **16**. The communication between the primary system processing blocks and the backup system processing blocks are illustrated by transmit and received packets **64**. The details of how transmit and receive packets **64** are formatted are not important for the purposes of this invention. The format will in large measure be determined by the details of backup transport link **16**. For example, if backup transport link **16** is a local area network then transmit and receive packet **64** will be formatted according to the conventions of the local area network. If backup transport link **16** is a dial-up connection using telephone lines, then the transmit and receive packet **64** will be any number of conventional communication protocols used to transfer data between computer systems over telephone lines. If backup transport link **16** is the internet, then transmit and receive packet **64** will be formatted according to one of the internet transfer protocols. Other connections may require other packet formats or communication protocols. All that is important for the current invention is that the data identified herein be able to be exchanged between the primary system and the backup system.

Referring now to FIG. **4**, one embodiment of mass storage read/write processing block **42** is presented. As previously

described, the function of mass storage read/write processing block **42** is to read data from or write data to mass storage device **20**. In addition, assuming that snapshot processing block **50** has been incorporated into read/write processing block **42**, then processing block **42** will also be responsible for preserving and maintaining a static snapshot of mass storage device **20** at a particular point in time. The implementation presented in FIG. **4** incorporates snapshot processing block **50** as an integral function. As previously described, however, it would also be possible to implement snapshot processing block **50** separately. The choice as to whether to incorporate snapshot processing block **50** into mass storage read/write processing block **42** or whether to implement snapshot processing block **50** separately is considered to be a design choice that is largely unimportant for purposes of the present invention. The important aspect for the present invention is to include the capability to read data from or write data to mass storage device **20** and the capability to preserve and maintain a static snapshot of at least a portion of mass storage device **20** at a particular point in time.

Turning now to FIG. **4**, decision block **66** first tests whether a snapshot request has been made. This decision block identifies whether the snapshot processing functionality incorporated into mass storage read/write processing block **42** should take a snapshot of at least a portion of mass storage device **20** of FIG. **3**. The snapshot request can come from the backup system or from another processing block of the primary system. Returning for a moment to FIG. **3**, a snapshot request is illustrated by snapshot request **68**. As illustrated in FIG. **3**, snapshot request **68** is generated by primary backup processing block **54**. As described in greater detail below, it is preferred that primary backup processing block **54** issue snapshot request **68**. Primary backup processing block **54** first identifies a logically consistent state before issuing such a snapshot request. In the alternative, the means for identifying a logically consistent state may be incorporated into the snapshot processing capability of mass storage read/write processing block **42** so that a snapshot request may be initiated either by the primary system or by the backup system and mass storage read/write processing block **42** would then identify a logically consistent state and take a snapshot. Such details are design choices and are not important from the point of view of this invention.

Returning now to FIG. **4**, if a snapshot request has been received, then the next step is to preserve a static snapshot of at least a portion of mass storage device **20**. Although any means to preserve a static snapshot can be used with the present invention, it is preferred that a particular process be used to preserve a static snapshot. The preferred method is summarized in the description of steps **70**, **72**, **74**, decision block **84**, and step **86** below. The method is more particularly described in U.S. Pat. No. 5,649,152 entitled METHOD AND SYSTEM FOR PROVIDING A STATIC SNAPSHOT OF DATA STORED ON A MASS STORAGE SYSTEM, previously incorporated by reference. In essence, a preferred method of preserving a static snapshot utilizes a snapshot storage, such as snapshot storage **22** of FIG. **3**, to preserve data blocks of a mass storage device, such as mass storage device **20** of FIG. **3**, that are to be overwritten with new data. As explained in greater detail below, the data blocks that are to be preserved are first copied into the snapshot storage and a record indicating that the data block has been preserved is updated. Such a record can be stored, for example, in snapshot map **52** of FIG. **3**. New data may then be written to mass storage device **20** without losing the preserved data blocks.

When a snapshot is to be taken, as evaluated by decision block **66**, the next step is to copy the snapshot map into the backup map as indicated by step **70** of FIG. **4**. As previously described, a backup map, such as backup map **48** of FIG. **3**, is used to indicate which data blocks have changed between a first instant in time and a second instant in time. These data blocks are then transferred to the backup system. As will become apparent in the description that follows, snapshot map **52** of FIG. **3** identifies those data blocks that have changed since a static snapshot was preserved at a particular instant in time. Thus, snapshot map **52** can be used as a backup map when a new snapshot is taken. Copying snapshot map **52** into a backup map **48** fulfills the desired function of identifying those data locations that have had new data stored therein between the time the last snapshot was taken and the current time. Obviously, it may not be necessary to copy the snapshot map to the backup map. The snapshot map may simply be used as the backup map and a new map taken as the current snapshot map.

After the snapshot map has been preserved so that it can be used as the backup map, the next step is to clear the current snapshot map. This step is indicated in FIG. **4** by step **72**. The snapshot map is used to store an indication of those data blocks that have had new data stored therein since the snapshot was taken. Thus, the snapshot map indicates which data blocks are stored in a snapshot storage, such as snapshot storage **22** of FIG. **3**. Since a new snapshot is to be taken, the snapshot map must be cleared.

After the snapshot map is cleared by step **72**, the next step is to clear snapshot storage, such as snapshot storage **22** of FIG. **3**. This is indicated by step **74** of FIG. **4**. With particular regard to this step, it should be noted that it may not be necessary to physically erase or clear the snapshot storage. Generally, as with any other type of storage, it is usually sufficient to clear the index into the storage to indicate that the storage is empty. Thus, if the index is kept as part of the snapshot storage map, such as snapshot storage map **52** of FIG. **3**, then clearing the snapshot storage map as performed in step **72** would be sufficient to indicate that the snapshot storage was empty. If, however, an index into the snapshot storage was kept separately from the snapshot storage map, then the index may need to be cleared separately by step **74**. After the snapshot map and snapshot storage have been cleared, the system is ready to preserve a new snapshot. Execution therefore precedes back to the start as indicated by FIG. **4**.

Attention is now directed to decision block **76** of FIG. **4**. This decision block tests whether a message received by mass storage read/write processing block **42** is a mass storage read or write request. This block is included in FIG. **4** simply to emphasize the fact that mass storage read/write processing block **42** only processes read or write requests to the mass storage device and a snapshot request as previously described. Decision block **76** may not be necessary as part of mass storage read/write processing block **42** as long as the only messages sent thereto are mass storage read and/or write requests.

By the time decision block **78** is reached, the only messages that are possible are either a mass storage read request or mass storage write request. This is because other types of requests are either handled or filtered out before decision block **78** is reached. Decision block **78** distinguishes between a mass storage read request and a mass storage write request. If a request is a mass storage read request, then the next step is to retrieve the requested data block from mass storage device **20** and return the data to the process making the request. This is illustrated in step **80**. If,

however, the request is a write request, then execution proceeds to decision block **82**.

Decision block **82** determines whether a snapshot is to be preserved. As previously described, in a preferred embodiment a snapshot is preserved by copying data blocks that are to be overwritten to a preservation memory such as snapshot storage **22** of FIG. **3**. In this embodiment, the snapshot is in essence preserved incrementally. In other words, when the snapshot is preserved, the snapshot storage is prepared to preserve data blocks as previously described in steps **72** and **74**. Thereafter, no data is stored in the snapshot storage until an actual write request occurs that will overwrite data that should be preserved. Thus, when a snapshot is preserved in this manner, it is important to determine if a snapshot has been taken or if write requests should occur to the mass storage system without worrying about preserving snapshot data. Decision block **82** tests whether the write request should occur without preserving snapshot data or whether snapshot data should be preserved for write requests. If the write requests should occur without preserving snapshot data, decision block **82** indicates that execution proceeds to step **88** where the data blocks are written to the mass storage device, such as mass storage device **20** of FIG. **3**. If, however, snapshot data should be preserved, then execution proceeds to decision block **84**.

As previously described, when a snapshot is taken according to a preferred embodiment, data which is to be overwritten is first copied to a snapshot storage, such as snapshot storage **22** of FIG. **3**. After the data has been preserved in the snapshot storage, the new data block can be written to the mass storage system. The goal of a snapshot is to preserve the data as it exists on the mass storage system at a particular point in time. Thus, the snapshot need only preserve the data as it existed at the time of the snapshot. Decision block **84** tests whether the original data block stored on the mass storage system at the time that the snapshot was taken has previously been preserved in the snapshot storage. In other words, if the data currently stored at the designated write storage location is data that was stored at that location at the moment in time when the snapshot was taken, then if the write request occurred without first preserving the data, the original data would be lost. If. however, the original data stored therein at the time the snapshot was taken has previously been preserved in the snapshot storage, then the write request may occur and overwrite whatever data is stored at the designated location without worry since the original data has previously been preserved. If, therefore, decision block **84** determines that the original data has not yet been stored in the snapshot storage, then execution proceeds to step **86**, which copies the original data into the snapshot storage. If, however, the original data has already been preserved, then step **86** is skipped.

After the original data has been preserved by step **86**, or a determination was made that the original data had previously been preserved, then execution proceeds to step **88** where the write request is filled by writing the data block included with the write request to the designated storage location on the mass storage device.

Step **90** then identifies the storage location as containing new data. As previously described, this may be accomplished by placing an entry in a snapshot map, such snapshot map **52** of FIG. **3**. Step **90** represents but one example of the previously described means for identifying storage locations of a mass storage device that have new data written therein.

A response may then be returned to the process making the write request. The sending of such a response is indicated in FIG. **4** by step **92**. Such responses are typically sent to the process that issues the write request not only to indicate the success or failure of the write operation but also to indicate completion of the write operation. Execution then proceeds back to the start where the next request is handled.

Turning next to FIG. **5**, the details of one embodiment implementing primary backup processing block **54** is presented. As previously described, primary backup processing block **54** is responsible for obtaining the data blocks that need to be transferred to the backup system and accomplishing the transfer using an appropriate communication protocol. As indicated in FIG. **5** by decision blocks **94** and **96**, primary backup processing block **54** first determines whether a backup has been initiated by the backup system or whether a backup should be initiated by the primary system. Primary backup processing block **54** will do nothing until a backup is either initiated by the backup system or by the primary system.

The present invention can be used in a variety of modes. As previously explained in one mode backups are initiated by the backup system. In such a system, the backup system may contact one or more primary systems and obtain the changes that have occurred since the last backup. Although such a mode can be used in a one-to-one situation, such a mode is extremely useful for what may be termed a many-to-one situation. In this mode, a centralized backup location may contact a plurality of primary systems located either locally or at remote sites and perform the backup for each contacted system in turn. In this mode, the backup system initiates the contact with one system, performs the backup, breaks the contact, and then initiates contact with the next system, and so forth. If simultaneous communication with multiple primary systems is available, the backup system may initiate contact with a number of primary systems at the same time. Using methods such as these, a company can backup critical data from anywhere in the world to a centralized location.

In another mode of operation, the backup is initiated by the primary system. In this mode of operation, the backup system waits for a primary system to establish contact and initiate a backup. The backup system then receives from the primary system the changes that have occurred since the last backup. In this mode, the backup system may also be acting to backup either a single primary system or a plurality of primary systems.

If the backup is initiated by the primary system as indicated in decision block **96**, then the primary system establishes a connection to the backup system as indicated in step **98**. This connection is established via backup transport link **16** of FIG. **3**. As indicated previously, backup transport link **16** may be any type of communication link that allows data to be transferred between the primary system and the backup system. Thus, step **98** will establish the connection using a method appropriate to the type of communication link between the primary system and the backup system. For example, if a dial-up connection is to be established, the primary system will dial the phone number of the backup system and establish contact using the appropriate communication protocol. Other connections are established using other types of protocols.

After the communication link has been established by the primary system if the backup is initiated by the primary system, or if decision block **94** detects that a backup has been initiated by the backup system, then execution proceeds to step **100**. Step **100** identifies a logically consistent backup state. As previously described, embodiments within

the scope of this invention may comprise means for identifying a logically consistent state of a mass storage device. Step **100** illustrates but one example of such means. Identifying a logically consistent state of the mass storage device may be accomplished either through an API or by monitoring activity on the mass storage device. Any method or mechanism that allows such a logically consistent state to be identified can be employed by the present invention.

After a logically consistent state has been identified, then a snapshot of the logically consistent state is preserved so that the backup may proceed. The snapshot is preserved by step **102** which signals the snapshot processing, as for example snapshot processing block **50** incorporated into a mass storage read/write processing block **42** of FIG. **3**, to take the snapshot. In one embodiment, this results in snapshot request **68** being sent to mass storage read/write processing block **42**. As previously described, this request will cause steps **70**, **72** and **74** of FIG. **4** to be executed, which prepares for the snapshot to be taken. Thereafter, original data stored in the mass storage device **20** at the time the snapshot was taken will be preserved by decision block **84** and step **86** of FIG. **4**.

After the snapshot has been taken in order to preserve the logically consistent backup state identified by step **100** of FIG. **5**, the next step in FIG. **5** is to assemble data blocks into a transmit packet as indicated by step **104**. As previously explained, backup transport link **16** of FIG. **3** may be implemented using a wide variety of technologies. In fact, several technologies may be used to communicate between a single backup system and a single primary system. For example, the two systems may be connected by a preferred backup transport link such as the internet or a high-speed, wide area network connection. If, however, the preferred link is unavailable then the system may revert to slower links such as a lower-speed dial-up connection. Thus, when step **104** indicates that data blocks should be assembled into a transmit packet, the format of the transmit packet will be dependent upon the exact communication link being used to send data between the backup system and the primary system. In some embodiments, depending upon the data block size and the transmit packet size, several data blocks may be able to be packed into a single transmit packet. In other situations, a single data block may need to be broken into several different transmit packets. Step **104** should be construed to include any translation or formatting that must occur to assemble a transmit packet for transfer to the backup system.

After the transmit packet has been assembled, step **106** sends the transmit packet to the backup system using an appropriate transmit protocol. After the transmit packet has been received by the backup system, step **108** tests whether more data remains to be sent. If so, execution proceeds back to step **104** where another transmit packet is assembled and sent. If no more data remains to be sent, then the connection to the backup system is terminated by step **110** and execution proceeds back to the start where primary backup processing block **54** waits until the next backup is initiated. The backups may be initiated, either by the backup system or by the primary systems, on a periodic schedule. Thus, the present invention may be used to capture a series of backups, each representing a logically consistent backup state, from one or more primary systems.

As previously described, the data blocks that are sent to the backup system by step **104** are only those data blocks that have changed since the last backup. Furthermore, the data blocks are transferred as they existed at the moment in time that the snapshot was taken. Thus, a backup map, such as backup map **48** of FIG. **3**, identifies the data blocks that should be transferred and the snapshot preserves those data blocks in the state that they were in when the snapshot was taken. Primary backup block **54** will therefore need to retrieve certain data blocks that were preserved by the snapshot. Primary backup processing block **54** may incorporate the functionality needed to retrieve the data blocks from the snapshot and/or mass storage system, or such functionality may be incorporated into a separate processing block. A separate processing block incorporating this functionality is illustrated in FIG. **3** by backup read processing block **56**. FIG. **6** presents one embodiment of backup read processing block **56** designed to recover the data preserved by these snapshots.

In FIG. **6**, decision block **112** highlights the fact that backup read processing block **56** only handles read requests that are to retrieve the data as it existed at the moment in time when the snapshot was taken. This decision block may not be necessary if the structure and architecture of the processing guarantees that only such read requests are sent to backup read processing block **56** of FIG. **3**.

In order to retrieve a data block as it existed at the moment in time when the snapshot was taken, it must be determined where the data block resides. As previously described in conjunction with FIG. **4**, after a snapshot is taken, the first time that a data block is to be overwritten by a new data block, the data block is copied into a snapshot storage, such as snapshot storage **22** of FIG. **3**. This means that if a data block is never overwritten, then the data stored on the mass storage device is the original data as it existed when the snapshot was taken. If, however, the data has been overwritten one or more times, then the original data will be stored in the snapshot storage. Decision block **114** of FIG. **6** determines whether the requested data block has been changed since the snapshot was taken. This may be accomplished by checking a snapshot map, such as snapshot map **52** of FIG. **3**, in order to determine whether the data block has been modified. As previously described, the snapshot map identifies those storage locations that have changed since the snapshot was taken.

If the storage location has had new data stored therein since the snapshot was taken, then step **116** indicates that the data block is retrieved from snapshot storage. If, however, the content of a storage location has not changed since the snapshot was taken, then step **118** indicates that the data block is retrieved from mass storage device **20**. In either case, the data block is returned to the requesting process by step **120**.

In order to illustrate in greater detail the operation of FIGS. **3–6** in creating a backup, a detailed example is presented in FIGS. **7A** and **7B**. Referring first to FIG. **7A**, consider a group of data blocks **122**, stored in storage locations numbered **1–6**, of mass storage device **20**. FIG. **7B** shows that backup storage device **24** also has a similar group of data blocks **124**, also stored in storage locations numbered **1–6**. At time T_0, the data blocks stored in **122** are identical to the data blocks stored in **124**. Referring again to FIG. **7A**, backup map **48** has six map locations **126** that correspond to storage locations **122**. Snapshot map **52** also has six map locations **128** that correspond to storage locations **122**. As illustrated in FIG. **7A**, at time T_0 map location **126** and **128** are cleared.

Assume that after time T_0, data blocks **130** are to be stored in locations **3** and **4** of storage locations **122**. One or more mass storage write requests will then be presented to mass storage read/write processing block **42** of FIG. **3** in order to

have data blocks **130** written to the appropriate storage locations. Turning to FIG. **4**, the mass storage write request would be processed in the following manner.

Decision blocks **66**, **76**, and **78** would combine to determine that a write request is being presented to mass storage read/write processing block **42**. Execution would thus pass through these three decision blocks to decision block **82**. As described previously, decision block **82** tests whether a snapshot has been taken. At this point in the example, no snapshot has been taken. Execution would thus proceed to step **88** which would write the requested data blocks into the mass storage system. Returning to FIG. **7A**, data blocks **130** would thus be stored in storage locations **122** to produce storage locations **132**. As indicated, therein, the data blocks stored in locations **3** and **4** have been modified to **3***a* and **4***a*.

Returning to FIG. **4**, step **90** next indicates that the storage locations where new data has been stored should be indicated as modified. In many snapshot embodiments, a snapshot map can be used for this purpose. In FIG. **7A**, map **134** is used and map locations **3** and **4** have been grayed to indicate that data has been stored in storage locations **3** and **4**. Note that the storage locations in backup map **48**, as indicated by map locations **126** remain unchanged at this point. Returning to FIG. **4**, a write request response would be returned by step **92** and execution would proceed back to the start to await the next request.

Returning now to FIG. **7A**, suppose that the next request contained three data blocks **136** that were to be stored in locations **3**, **4**, and **6**. Since a snapshot has not yet been taken, this request will be handled in the same way as the previous write request with execution proceeding through decision blocks **66**, **76**, **78**, and **82** of FIG. **4** to step **88** of FIG. **4**. Step **88** indicates that the new data is stored in the mass storage device so that storage locations **138** of FIG. **7A** now indicate that the data blocks stored in location has been changed to **3***b*, the data block stored in location **4** has been changed to **4***b*, and the data block stored in location **6** has been changed to **6***a*. As with the previous write request, map locations **140** are then updated to indicate that in addition to locations **3** and **4**, location **6** has also been changed. Map locations **126** remain unchanged.

Assume at this point in our example that the backup system or the primary system initiates a backup. Primary backup processing block **54** of FIG. **3** will then begin executing as described in FIG. **5**. In FIG. **5**, if the backup was initiated by the backup system, execution would proceed from decision block **94** to step **100**. If, however, the backup was initiated by the primary system, then execution would proceed from decision block **96** to step **98** where a connection would be established to the backup system. In any event, execution would proceed to step **100**. In step **100**, primary backup processing block **54** would identify a logically consistent backup state. As previously explained, this may be accomplished in any way such as, for example, watching the activity on mass storage device **20** or through an API.

After identifying a logically consistent backup state, step **102** indicates that the signal to take a snapshot is sent. As previously described, rather than signalling a snapshot to be taken, the means to preserve a static snapshot may be incorporated directly into step **102**. In the embodiment illustrated in FIG. **3**, and described in greater detail in FIGS. **4–6**, step **102** would send snapshot request **68** of FIG. **3** to mass storage read/write processing block **42**.

Turning now to FIG. **4**, this snapshot request will be processed by decision block **66** which will result in steps **70**, **72**, and **74** being executed. In step **70**, the snapshot map is copied to the backup map. In FIG. **7A**, this means that map locations **140** are copied into map locations **142** of backup map **48**. Thus, map locations **142** indicate that locations **3**, **4**, and **6** have had new data stored therein. Returning now to FIG. **4**, step **72** clears the snapshot map and step **74** clears the snapshot storage as previously described. Execution in FIG. **4** would then return to the start to await further processing.

Assume at this point, that a write request arrived at mass storage read/write processing block **42** requesting that data blocks **144** of FIG. **7A** be stored in storage locations **138**. Because this is a write request, execution will proceed through decision blocks **66**, **76**, and **78** to decision block **82**. Unlike previous write requests, a snapshot has now been taken at time T_1 as indicated in FIGS. **7A** and **7B**. Thus, execution will proceed to decision block **84**.

Decision block **84** determines whether the data stored in the storage locations that are to be overwritten have been previously stored in snapshot storage. In this example, data blocks **144** are to be stored in storage locations **1** and **3**. Since storage locations **1** and **3** have not yet been placed in snapshot storage, step **86** will copy storage locations **1** and **3** into snapshot storage. In FIG. **7A**, this is illustrated by data block **146** containing data block **1** and data block **148** containing data block **3***b*.

After data block **146** and **148** have been preserved in snapshot storage **22**, the new data blocks are written to the mass storage device by step **88**. Returning to FIG. **7A**, this means that data blocks **144** are stored in storage locations **138** in order to produce storage locations **150** where data block 1a has overwritten data block **1** and data block **3***c* has overwritten data block **3***b*. Step **90** of FIG. **4** then states that the data blocks need to be identified as modified. Thus, map locations **152** of snapshot map **52** are modified to indicate that storage location **1** and storage location **3** have new data stored therein. A write request response is then returned as directed by step **92** of FIG. **4**.

Returning now to FIG. **5**, the snapshot was taken at time T_1 by mass storage read/write processing block **42** of FIG. **3** as directed by step **102** of FIG. **5**. Step **104**, step **106**, and decision block **108** then indicate that the data blocks that were changed before the snapshot was taken should then be assembled into transmit packets and sent to the backup system. The data blocks that should be transferred are indicated by the information contained in backup map **48**.

Returning to FIG. **7A**, map locations **142** of backup map **48** indicate that storage locations **3**, **4**, and **6** have been changed prior to the snapshot taken at time T_1 and should be sent to the backup system. An examination of storage locations **150** and data block **148** stored in snapshot storage **22** indicates that one of the data blocks is in the snapshot storage while the remainder of the data block are on the mass storage system. Step **104** of FIG. **5** would then request that data blocks stored in storage locations **3**, **4**, and **6** be retrieved by backup read processing block **56** of FIG. **3**.

Backup read processing block **56** will process these requests received from primary backup processing block **54** as illustrated in FIG. **6**. The request will be for the data blocks stored in storage locations **3**, **4**, and **6**. With regard to the data block stored in storage location **3**, decision block **114** of FIG. **6** will identify that the decision block stored in storage location **3** has changed since the snapshot was taken. This is because the data block labeled **3***c* was stored in storage location **3** after the snapshot was taken, but before the data block was retrieved for the backup. Step **116** will

then retrieve data block **148** from snapshot storage **22** and return data block **3***b* to primary backup processing block **54** as illustrated in step **120** of FIG. **6**.

Decision block **114** of FIG. **6** will then retrieve the data block stored in storage locations **4** and **6** from the mass storage device in step **118** and return them to primary backup processing block **54** in step **120**. This process is illustrated graphically in FIG. **7**A where data blocks **152** are assembled by retrieving data block **3***b* from snapshot storage **22** and data block **4***b* and **6***a* from storage locations **150**. Data blocks **152** are then transferred to the backup system, via backup transport link **16**. This is graphically illustrated in FIGS. **7**A and **7**B.

As described in greater detail below, when data blocks are received by a backup system it may be desirable to store the data blocks as they are received in a backup capture buffer, such as backup capture buffer **26** of FIG. **3**. This allows all data blocks to be received before they are applied to backup storage device **24** or before they are saved as an incremental backup. In FIG. **7**B, data blocks **152** are received by the backup system and applied to storage locations **124** to achieve storage locations **154**. Storage locations **154** are identical to storage locations **138** of the primary system (FIG. **7**A). Recall that storage locations **138** represented the state of mass storage device **20** at time T_1 when the snapshot was taken. Thus, the changes that have occurred between time T_0 and time T_1 have now been backed up to the backup system and applied to backup storage device **24** in order to bring backup storage device **24** current with mass storage device **20** at time T_1.

Returning now to FIG. **7**A, suppose that data blocks **156** are now to be written to storage locations **150**. As illustrated therein, data blocks **156** comprise a change to the data blocks stored in storage locations **1**, **4**, and **6**. Mass storage read/write processing block **42** will handle the write of the data blocks to be stored in locations **4** and **6** as previously described with the original data blocks stored in those locations at time T_1 (data block **4***b* and data block **6***a*) being stored in snapshot storage **22**. New data blocks **4***c* and **6***b* will then be written to mass storage device **20**.

With regard to the data block that is to be stored in storage location **1**, execution will proceed in FIG. **4** down to decision block **84**. Recall this decision block tests whether the data block stored in the storage location at the time that the snapshot was taken has previously been preserved in the snapshot storage. With regard to the data block stored in storage location **1**, the data block has been previously preserved in snapshot storage **22** as indicated by data block **146** of FIG. **7**A. Thus, FIG. **4** indicates that step **86** is skipped and the new data is simply written to the mass storage device. In FIG. **7**A, this results in data block **1***b* replacing data block **1***a* so that data block **1***a* is lost.

Recall that the present invention only transfers the data blocks of those storage locations that have changed since the last backup. Furthermore, the data blocks are transferred as they exist at the time that the snapshot is made. Thus, if a particular storage location on the mass storage device has five different data blocks stored therein during the time since the last backup, only the data block stored last (e.g. just before the snapshot is taken) is transferred to the backup system. This is because the backup system only preserves a logically consistent backup when the backup is taken. In other words, the backup storage moves from a logically consistent state at one moment in time to a logically consistent state at another moment in time. Preserving logically consistent backups at discrete moments in time provides significant advantages over prior art systems.

For example, consider a prior art system that captures each and every change made to a mass storage system. Such a prior art system will attempt to send every write operation both to the mass storage device and to the backup storage device. In theory, this makes the backup storage device an identical copy of the mass storage device. However, problems arise with this approach. If the primary system crashes during a write update, it may leave the mass storage device in a logically inconsistent state. If the backup storage device is tracking every change made to the mass storage device, then when the primary system crashes, the backup storage device may also be left in the same logically inconsistent state. This example highlights the problem of leaving a known logically consistent state before a second logically consistent state has been identified. The present invention avoids this problem by maintaining the prior logically consistent state until a new logically consistent state has been identified and then moves the backup storage device from the previous logically consistent state to the next logically consistent state without transitioning through any logically inconsistent states between the two logically consistent states.

Returning to FIG. **7**A, when data blocks **156** are applied to storage locations **150**, storage locations **158** result. Map locations **152** are then updated to indicate that the storage locations that have been changed since time T_1 now include storage locations **4** and **6** in addition to storage locations **1** and **3**. This is illustrated in FIG. **7**A by map locations **160** of snapshot storage **152**.

Assume that a second backup is now to be made of mass storage device **20**. The backup will be made as previously described in FIG. **5**, where execution proceeds to step **100** where a logically consistent state is identified. In FIG. **7**A, assume this logically consistent state was identified at time T_2. Step **102** of FIG. **5** would then signal a snapshot to be taken at time T_2. As previously described in conjunction with the snapshot taken at time T_1, mass storage read/write processing block **42** would receive a snapshot request, such as snapshot request **68** of FIG. **3**, and will copy the snapshot map to the backup map in step **70**. This is indicated in FIG. **7**A where map locations **162** of backup map **48** are changed to be the same as map locations **160** of snapshot map **52**.

Steps **72** and **74** of FIG. **4** then indicates that the snapshot map and snapshot storage should be cleared. In FIG. **7**A, the snapshot map is cleared as indicated by map locations **164** of snapshot map **52**. Snapshot storage **22**, however, still shows data blocks stored therein. This is to illustrate that the data blocks may still physically reside in snapshot storage **22** as long as the index to snapshot storage **22** is cleared so that snapshot storage **22** appears to contain no data blocks.

Assuming that no data blocks are within storage locations **158** after the snapshot taken at time T_2, then data blocks **166** will be read from storage locations **158** according to the process described in FIG. **6**. The data blocks will then be packaged into transmit packets and sent to the backup system via backup transport link **16** as illustrated in step **104**. step **106**, and decision block **108** of FIG. **5**. As illustrated in FIG. **7**B data blocks **166** will then be stored in backup snapshot buffer **26** until all data blocks are received. After data blocks **166** have been received by the backup system, then are applied to storage locations **154** in order to arrive at storage locations **168**, which are an identical copy of storage locations **158** of the primary system (FIG. **7**A).

The mechanism to discover differences between mass storage device **20** and backup storage device **24** is described next. Embodiments within the present invention may com-

prise means for identifying differences between a mass storage device and a backup storage device. Such means may be very useful in recovering from crashes that happen on the primary or backup system. For example, it is apparent from the previous example and above descriptions that the present invention tracks changes made to the mass storage device between a first instant in time, such as the time that the last backup was made, to a second instant in time, such as the time that a current backup is to be made. If a backup is to be made at the second instant in time, the primary system then preserves a snapshot of at least the storage locations that have had new data written therein. The data blocks are then retrieved and transferred to the backup system. During the transfer process, the system also tracks changes that are made so that when another backup is to be made, all the changes from the last backup to the current backup can be identified.

The above described process works very well as long as there is not an interruption in tracking changes that are made to the mass storage device. If, however, a situation arises where the primary system cannot identify which changes have been made to the mass storage system since the last backup, then a mechanism must be in place for identifying differences between the mass storage device and the backup storage device. By identifying differences between the mass storage device and the backup storage device, those storage locations that are different can be identified. The data stored in those storage locations can then be transferred from the primary system to the backup system in order to bring the backup system current with the primary system.

One mechanism to identify differences between a mass storage device and a backup storage device is to compare each and every data block on the mass storage device and the backup storage device. This requires transferring either the data blocks of the mass storage device to the backup system or transferring the data blocks of the backup storage device to the primary system. In certain circumstances this may be entirely adequate. However, this method requires a fairly large bandwidth for backup transport link **16**. If, however, backup transport link **16** is a relatively low bandwidth communication link, then transporting each and every data block of either the mass storage device or the backup storage device across backup transport link **16** becomes impractical. In such a situation, a mechanism must be in place to reduce the amount of data that is transferred across backup transport link **16**.

In order to reduce the amount of data needed to identify differences between a mass storage device and a backup storage device, embodiments within the scope of this invention may comprise means for calculating a digest from a data block. As used herein, a "digest" is a group of data bits that is generated from a data block and that reflects the data block. If the digest is smaller than a data block and if the digest reflects the data of a data block, then differences between a mass storage device and a backup storage can be identified by comparing digests. Such a method is illustrated in FIG. **8**.

In FIG. **8**, the method to identify differences between a mass storage device, as for example mass storage device **20**, and a backup storage device, as for example backup storage device **24**, proceeds as follows. The backup system retrieves a data block, such as data block **170** from backup storage device **24**. Digest **172** is then calculated by means for generating a digest, as for example digest generation block **174**. Digest **172** is transported across backup transport link **16** and received by the primary system. The primary system retrieves a data block stored in the corresponding storage location of mass storage **20**, as for example data block **176**. Digest **178** is generated by digest generation means, as for example digest generation block **180**. Digest **178** is then compared to received digest **172** by compare block **182**. If the digests match, then the data blocks stored in the corresponding storage locations can be assumed to be identical. If the digests do not match, then the data blocks stored in the corresponding storage locations are different. A similar mechanism can be used to detect differences in groups of data blocks. For example, a digest can be calculated on a plurality of concatenated data blocks. Differences in the digests would then identify differences in groups of data blocks.

From the above description of FIG. **8**, several desirable properties of the digest can be identified. Ideally, a digest would be small in length to minimize the amount of data that needs to be transferred between the primary system and the backup system. Second, the probability of two different data blocks generating the same digest should be small so that the probability of identifying two different data blocks to be the same when they are actually different is small. Finally, it would be desirable, although not required, to reduce the computation burden needed to calculate the digest so that the process of comparing the mass storage device to the backup storage device is limited not by the computations performed by the primary system or the backup system but, rather, by the bandwidth of backup transport link **16**.

A wide variety of functions can be used to calculate an appropriate digest. The simplest, and perhaps most well known form of digest, is a cyclic redundancy check or CRC. CRC values are typically used to detect errors in a block of data transmitted across a communication link or stored on a storage device. Cryptographically strong hash functions (also referred to as digests, fingerprints, or message authentication codes) have also been developed to perform a similar function. Any method can be used as long as the digest has a sufficiently high probability of detecting differences between two data blocks.

As previously described, difference identification processing block **58** of FIG. **3** is used to identify differences between mass storage device **20** and backup storage device **24**. Turning now to FIG. **9**, the details of one embodiment implementing difference identification processing block **58** are presented. In this embodiment, it is presumed that the digests are transferred from the backup system to the primary system and the primary system compares the digests to determine if they match.

As illustrated by step **184** of FIG. **9**, the first step is to establish a connection to the backup system. The first data block of mass storage device **20**, or the first data block of the portion of mass storage device **20** that is to be checked, is then retrieved by step **186**. The digest is then calculated for the data block by step **188**.

In step **190**, the digest calculated by the backup system on the data stored in the corresponding storage location is received. The digests are then compared by decision block **192**. If the digests do not match, then it may be presumed that the data blocks do not match and the data block on mass storage device **20** can be presumed to have changed since the backup captured by backup storage device **24**. This data block of mass storage device **20** is then identified in step **194** as changed since the last backup. Returning now to FIG. **3**, if the difference identification block **58** is used to rebuild snapshot map **52** after a crash, then difference identification block **58** can store the results of the compare in snapshot map **52** or backup map **48**.

If more data blocks exist to be compared, decision block **196** and step **198** retrieve the next data block and return execution to step **188** where the digest is calculated on the next data block. If no more data blocks need to be compared, step **200** terminates the connection to the backup system and the compare process is complete.

It is apparent that if the digest is much smaller than a data block or a group of data blocks, then the amount of data that needs to be transferred between the primary system and the backup system in order to identify differences between the mass storage device and the backup storage device can be greatly minimized. For example, if a data block is 512 bytes long and a digest is two bytes long, then the data transferred can be reduced by a factor of 256. This can result in a significant time savings. It also makes it feasible to compare a backup storage device located at a remote site to a mass storage device using only a relatively low bandwidth dial-up communication link.

Turning now to FIG. **10**, the processing of one embodiment of backup system processing block **60** of FIG. **3** is presented. This is the processing that occurs on the backup system. The processing illustrated in FIG. **10** is straightforward given the previous discussion and represents the complimentary processing to primary backup processing block **54** and difference identification processing block **58**.

Decision block **202** of FIG. **10** identifies whether difference identification processing block **58** is attempting to compare the differences between mass storage device **20** and backup storage device **24**. If the differences are to be identified, then execution proceeds to step **204** where the first data block of the last known backup state is retrieved. Step **206** then calculates the digest for this data block and step **208** transfers the digest to the primary system. This digest is received by difference identification processing block **58** at step **190** of FIG. **9**. Decision block **210** in step **212** then tests whether more data blocks exist and if so, retrieves the next data block and then returns processing to step **206** so that the digest for that data block can be calculated. When all data blocks have been processed, execution returns back to the start.

Decisions blocks **214** and **216** of FIG. **10** identify whether a backup is being initiated by either the backup system or by the primary system. The decision blocks are analogous to decision blocks **94** and **96** of FIG. **5**. If the backup is to be initiated by the backup system, then a connection is established to the primary system by step **218**. If the backup is initiated by the primary system, then the connection will have previously been established and execution can proceed directly to step **220**.

Step **220** receives a packet from the primary system using the appropriate protocol. This packet would be transferred to the backup system by step **106** of FIG. **5**. The packet will contain one or more data blocks or portions thereof, depending on the size of the data block with respect to the size and format of a packet. Step **222** then buffers the received data blocks in a backup capture means such as backup capture buffer **26**.

Buffering received data until all data blocks have been received is an important step in the present invention. As emphasized throughout this application, the present invention transfers the data blocks stored only in those storage locations that have had new data stored therein since the last backup. Furthermore, the data transferred is the data that is stored in those locations at the time that the snapshot is made. Thus, the time sequence of changes is not transferred and only the ultimate result of all the time sequence of changes since the last backup is transferred. This means that applying only a portion of the data blocks that are to be transferred may result in a logically inconsistent backup. It is, therefore, undesirable to apply only a portion of the data blocks that are to be transferred between the primary system and the backup system for a single backup. If the data blocks are applied to the backup storage device as they are received, and if the backup system or primary system crashes during the transfer, then the backup storage device may be left in a logically inconsistent state. For these reasons, it is presently preferred that the received data blocks be buffered in a temporary location until all data blocks are received. The data blocks may then be applied to the backup storage device or may be saved as an incremental backup all at once.

Decision block **224** ensures that all data blocks have been received. Once all data blocks have been received, decision block **226** tests whether a complete set has been received. This complete set comprises all the changes that have occurred since the last backup. If a complete set has not been received, then appropriate actions should be taken. For example, step **228** illustrates that the backup set should be discarded and not applied to the mass storage device. This is important for the reasons previously discussed. However, before a backup set is discarded additional efforts to recover any changes that are missing from the set may be undertaken. For example, the backup system can initiate contact with the primary system and request transfer of the missing changes. If the primary system is currently unavailable, then perhaps the partial set may be stored separately until contact with the primary system can be re-established. At that point, the backup system can inform the primary system which changes have been received and which changes remain to be transferred. Such attempts at recovery of missing changes may reduce the amount of data that needs to be retransferred between the primary system and the backup system. If, however, the changes cannot be recovered, then the entire set should be discarded and a new set received from the primary system.

If a complete set of changes have been received, then step **230** indicates that the changes should be processed using the desired method. Throughout this description, reference has been made to applying a group of changes to the backup storage device in order to bring the state of the backup storage device current to a particular point in time. In addition to applying the changes in this manner, the changes may also be saved as an incremental backup. By saving the changes as an incremental backup, several past backup states may be stored. This way, should data need to be recovered from the backup system, several backup states are available to choose from. Combinations of the above may also be used. For example, several incremental backups may be kept at which time the incremental backups are applied to an initial state in order to bring the backup device current to a particular point in time.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrated and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed and desired to be secured by United States Letters Patent is:

1. In a computing environment including a primary system having a primary mass storage device that stores a plurality of primary data blocks and a backup system having

a backup mass storage device that stores a plurality of backup data blocks, a method for comparing the primary mass storage device to the backup mass storage device, comprising the steps of:

calculating, by the backup system, a first digest based on a selected backup data block, the selected backup data block corresponding to a physical location within the backup mass storage device, wherein the first digest is smaller than the selected backup data block;

calculating, by the primary system, a second digest based on a selected primary data block, the selected primary data block corresponding to the selected backup data block and also corresponding to a physical location within the primary mass storage device, wherein the second digest is smaller than the selected primary data block, and

comparing the second digest with the first digest to determine whether the second digest and the first digest indicate that the selected backup data block and the selected primary data block contain the same data.

2. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1** wherein the first and second digests represent data of the respective selected primary and backup data blocks, the digests having a high probability of uniquely correlating to the data.

3. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1**, further comprising, prior to the step of calculating the second digest, the step of transmitting the first digest from the backup system to the primary system.

4. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **3**, wherein the step of comparing is conducted at the primary system.

5. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1**, wherein the step of comparing is conducted at the backup system.

6. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1** wherein the step of comparing the second digest with the first digest comprises the step of determining whether the second digest and the first digest are the same.

7. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1**, further comprising the steps of:

establishing a connection between the primary system and backup system; and

terminating the connection between the primary system and backup system after the step of comparing has been completed.

8. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1**, further comprising the step of storing, at the primary system, the results of the comparing step.

9. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1**, further comprising repeating the steps of retrieving a selected backup data block, calculating a first digest, transmitting the first digest, retrieving a selected primary data block, calculating a second digest, and comparing the second digest with the first digest until all of the plurality of primary data blocks have been compared to corresponding backup data blocks.

10. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1** wherein the step of comparing comprises the step of determining that the selected backup data block and the selected primary data block do not contain the same data, and wherein the method further comprises the step of updating the selected backup data block so that it contains the same data as the selected primary data block.

11. In a computing environment including a primary system having a primary mass storage device that stores a plurality of primary data blocks and a backup system having a backup mass storage device that stores a plurality of backup data blocks, a method for comparing the primary mass storage device to the backup mass storage device, comprising repeating, until all primary data blocks have been compared to corresponding backup data blocks, the steps of:

calculating, by the backup system, a first digest based on a selected backup data block, the selected backup data block corresponding to a physical location within the backup mass storage device, wherein the first digest is smaller than the selected backup data block;

transmitting the first digest from the backup system to the primary system;

calculating, by the primary system, a second digest based on a selected primary data block, the selected primary data block corresponding to the selected backup data block and also corresponding to a physical location within the primary mass storage device, wherein the second digest is smaller than the selected primary data block; and

comparing the second digest with the first digest to determine whether the second digest and the first digest indicate that the selected backup data block and the selected primary data block contain the same data.

12. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **1**, further comprising the step of updating the selected backup data block so that it contains the same data as the selected primary data block when the step of comparing the second digest and the first digest indicates the data blocks contain different data.

13. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **11** wherein the method is initiated by the primary system.

14. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **11** wherein the method is initiated by the backup system.

15. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **11** wherein the backup system is remotely located from the primary system.

16. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **11** wherein the first and second digests represent multiple data blocks.

17. A method for comparing the primary mass storage device to the backup mass storage device as recited in claim **11** wherein calculating the first digest is computationally faster than transmitting the first digest.

18. In a computing environment including a mass storage device having a plurality of data blocks, a method for comparing data blocks, comprising the steps of:

selecting a first data block;

calculating a first digest, the first digest being generated from data contained within the first data block and being smaller than the first data block, the first data block corresponding to a physical location within the mass storage device;

selecting a second data block;

calculating a second digest, the second digest being generated from data contained within the second data block and being smaller than the second data block, the second data block corresponding to a physical location within the mass storage device, the first and second digests being calculated in a process wherein a change in the data blocks would have a high probability of changing the value of the corresponding digest; and

comparing the first digest with the second digest to determine whether the first digest and the second digest indicate that the first data block and the second data block contain the same data.

19. A system for comparing mass storage, comprising:

primary mass storage means attached to a primary system for storing a plurality of primary data blocks;

backup mass storage means attached to a backup system for storing a plurality of backup data blocks;

backup system processor means for performing the steps of:

retrieving a backup data block from the backup mass storage means, the backup data block corresponding to a physical location within the backup mass storage means;

calculating a first digest based on data stored in the backup data block, the first digest being smaller than and having a high probability of uniquely correlating to the backup data block it represents; and

repeating the above steps until all backup data blocks have been considered; and

primary system processor means for performing the steps of:

receiving the first digest from the backup system processor means;

retrieving a primary data block from the primary mass storage means, the primary data block corresponding to a physical location within the primary mass storage means and also corresponding to the backup data block associated with the first digest;

calculating a second digest based on data stored in the primary data block, the second digest being smaller than and having a high probability of uniquely correlating to the primary data block it represents;

comparing the first digest and the second digest;

interpreting any difference between the first and second digests to indicate a difference between the primary mass storage means and the backup mass storage means; and

repeating the above steps until all primary and backup data blocks have been considered.

20. A system for comparing mass storage as recited in claim **19** further comprising:

primary processor means for performing the steps of:

establishing a connection between the primary processor means and backup processor means;

storing the results of the comparing step; and

transmitting an update data block to the backup processor means, when the comparing step indicates that the primary data block and the backup data block contain different data; and

backup processor means for performing the steps of:

receiving the update data block from the primary processor means; and

updating the backup data block with the received update data block.

21. A system for comparing mass storage as recited in claim **19**, wherein the backup system processor means is remotely located from the primary system processor means.

22. A system for comparing mass storage as recited in claim **19**, wherein the first and second digests represent multiple data blocks.

23. A computer program product for implementing a method for use in a backup system including a backup mass storage device that stores a plurality of backup data blocks, the backup system being connected to a primary system including a primary mass storage device that stores a plurality of primary data blocks, the computer program product comprising:

a computer-readable medium carrying computer-executable instructions for implementing the method, wherein the computer-executable instructions comprise:

means for retrieving a backup data block from the backup mass storage device, the backup data block corresponding to a physical location within the backup mass storage device;

means for calculating a first digest based on data stored in the backup data block, the first digest being smaller than and having a high probability of uniquely correlating to the backup data block it represents; and

means for transmitting the first digest to the primary system;

wherein retrieving the backup data block, calculating the first digest, and transmitting the first digest are conducted until all of the plurality of backup data blocks have been considered.

24. A computer program product as recited in claim **23**, wherein the computer-executable instructions further comprise backup means for receiving an update data block from the primary system for updating a corresponding backup data block when it has been determined that the corresponding backup data block and a corresponding primary data block at the primary mass storage device contain different data.

25. A computer program product as recited in claim **23**, wherein the means for calculating a first digest calculates the first digest based on data stored in multiple backup data blocks.

26. A computer program product for implementing a method for use in a primary system including a primary mass storage device that stores a plurality of primary data blocks, the primary system connected to a backup system including a backup mass storage device that stores a plurality of backup data blocks, the computer program product comprising:

a computer-readable medium carrying computer-executable instructions for implementing the method, wherein the computer-executable instructions comprise:

means for receiving a first digest from the backup system;

means for retrieving a primary data block from the primary mass storage device that corresponds to the first digest, the primary data block corresponding to a physical location within the primary mass storage device;

means for calculating a second digest based on data stored in the primary data block, the second digest being smaller than and having a high probability of uniquely correlating to the primary data block it represents;

means for comparing the first digest and the second digest; and

means for interpreting any difference between the first and second digests to indicate a difference between

the backup mass storage device and the primary mass storage device;

wherein receiving the first digest, retrieving the primary data block, calculating the second digest, comparing the first digest and the second digest, and interpreting any difference are conducted until all backup data blocks and all primary data blocks have been considered.

27. A computer program product as recited in claim **26**, wherein the computer executable instructions further comprise means for transmitting an update data block to the backup system for updating the corresponding data block in the backup system, wherein a difference between the first and second digests has indicated a difference between the backup mass storage device and the primary mass storage device.

28. A computer program product as recited in claim **26**, wherein the means for calculating a second digest calculates the second digest based on data stored in multiple primary data blocks.

* * * * *

US006065018A

United States Patent [19]

Beier et al.

[11] Patent Number: 6,065,018

[45] Date of Patent: May 16, 2000

[54] **SYNCHRONIZING RECOVERY LOG HAVING TIME STAMP TO A REMOTE SITE FOR DISASTER RECOVERY OF A PRIMARY DATABASE HAVING RELATED HIERARCHIAL AND RELATIONAL DATABASES**

[75] Inventors: **Harley Al Beier**, San Martin, Calif.; **Robert Frederic Kern**, Southampton, United Kingdom; **David Wayne Moore**, Morgan Hills, Calif.; **Karen Alicia Ranson**, San Jose, Calif.; **Vern Lee Watts**, Los Altos, Calif.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **09/034,867**

[22] Filed: **Mar. 4, 1998**

[51] **Int. Cl.**[7] **G06F 17/30**

[52] **U.S. Cl.** **707/202**; 707/201; 707/203; 707/204; 714/12; 714/20; 709/101; 709/201

[58] **Field of Search** 707/1–5, 8, 10, 707/200–205, 100–104; 714/1, 6, 12–13, 20, 731, 744, 775, 16–18; 711/113, 117–118, 130, 147, 153, 161, 162, 167–168; 709/101, 201, 203, 212–215, 217, 232, 248

[56] **References Cited**

U.S. PATENT DOCUMENTS

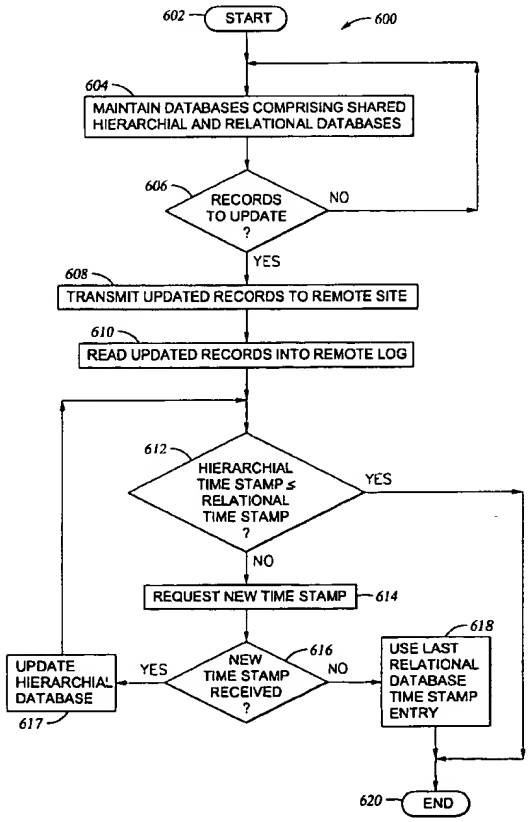
5,170,480	12/1992	Mohan et al.	707/201
5,278,982	1/1994	Daniels et al.	707/202
5,446,871	8/1995	Shomler et al.	714/1
5,561,795	10/1996	Sarkar	707/202
5,566,297	10/1996	Devarakonda et al.	714/15
5,594,863	1/1997	Stiles	714/15
5,594,900	1/1997	Cohn et al.	707/202
5,615,329	3/1997	Kern et al.	714/6
5,619,644	4/1997	Crockett et al.	714/45
5,640,561	6/1997	Satoh et al.	707/202
5,673,382	9/1997	Cannon et al.	714/6
5,682,513	10/1997	Caldelaria et al.	711/113
5,852,715	12/1998	Raz et al.	709/201
5,913,219	6/1999	Back et al.	707/202
5,924,096	7/1999	Draper et al.	707/10
5,949,970	9/1999	Sipple et al.	395/182.13

Primary Examiner—Anton W. Fetting
Assistant Examiner—Srirama Channavajjala
Attorney, Agent, or Firm—Gray Cary Ware Freidenrich

[57] **ABSTRACT**

A method and apparatus to synchronize recovery logs transmitted to a remote site for recovering related databases having different logical structuring. In one embodiment, the related databases are a hierarchial structured database such as IMS and a relational structured database such as DB2.

27 Claims, 7 Drawing Sheets



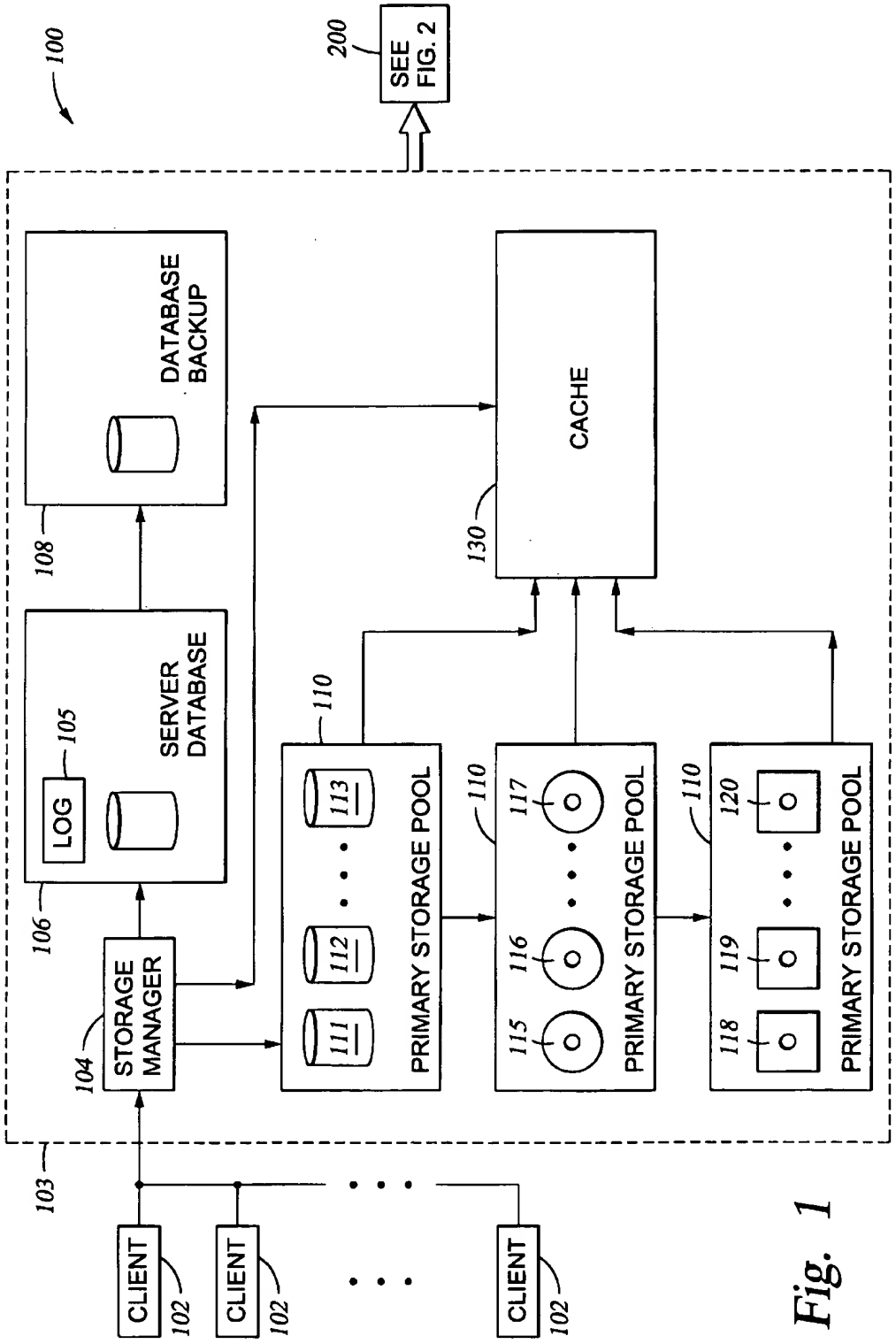

100
200
SEE FIG. 2
DATABASE BACKUP
108
CACHE
130
SERVER DATABASE
LOG
105
106
110
PRIMARY STORAGE POOL
111
112
113
117
116
115
120
119
118
STORAGE MANAGER
104
103
CLIENT
102

Fig. 1

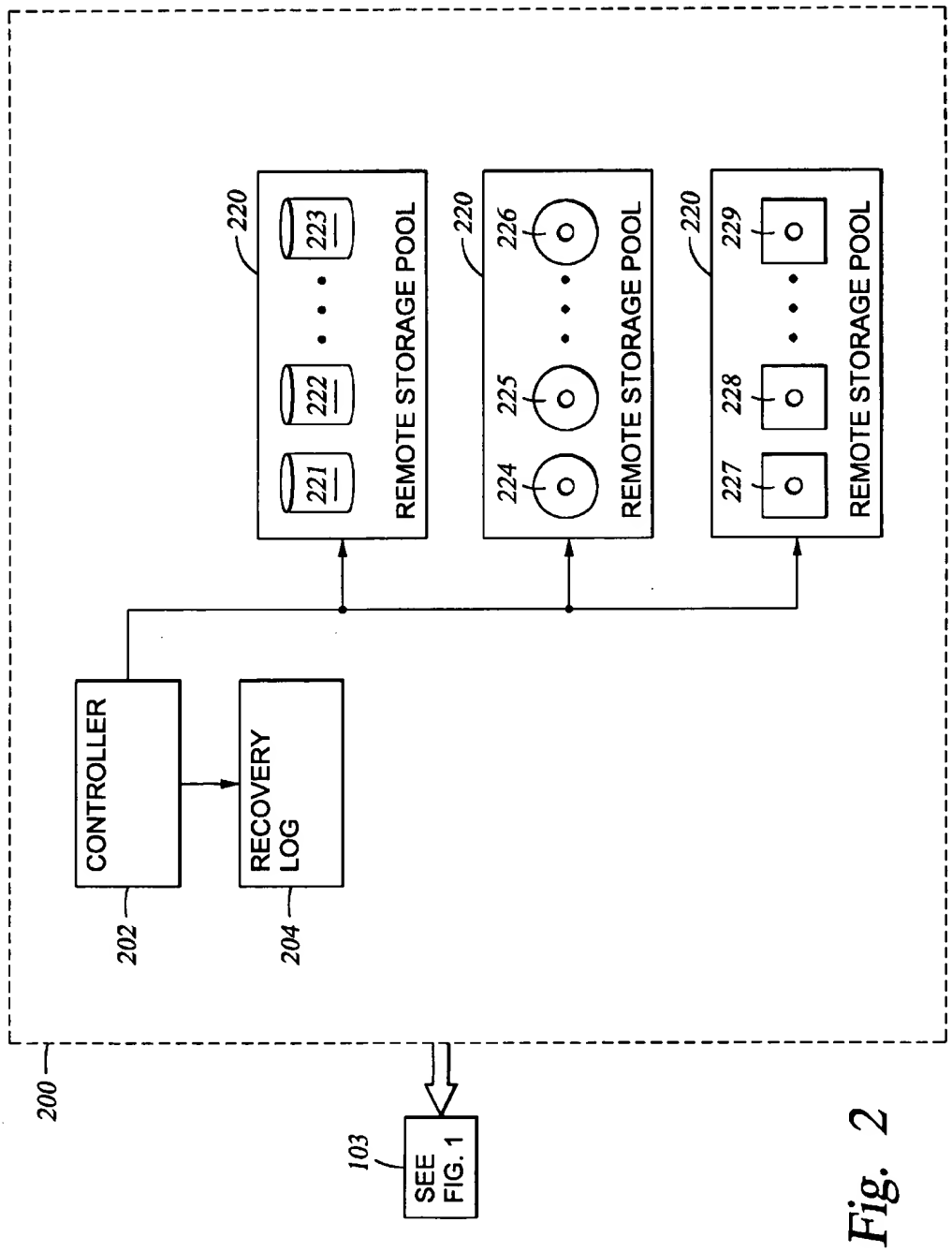

200
CONTROLLER
202
RECOVERY LOG
204
220
221
222
223
REMOTE STORAGE POOL
220
224
225
226
REMOTE STORAGE POOL
220
227
228
229
REMOTE STORAGE POOL
103
SEE FIG. 1

Fig. 2

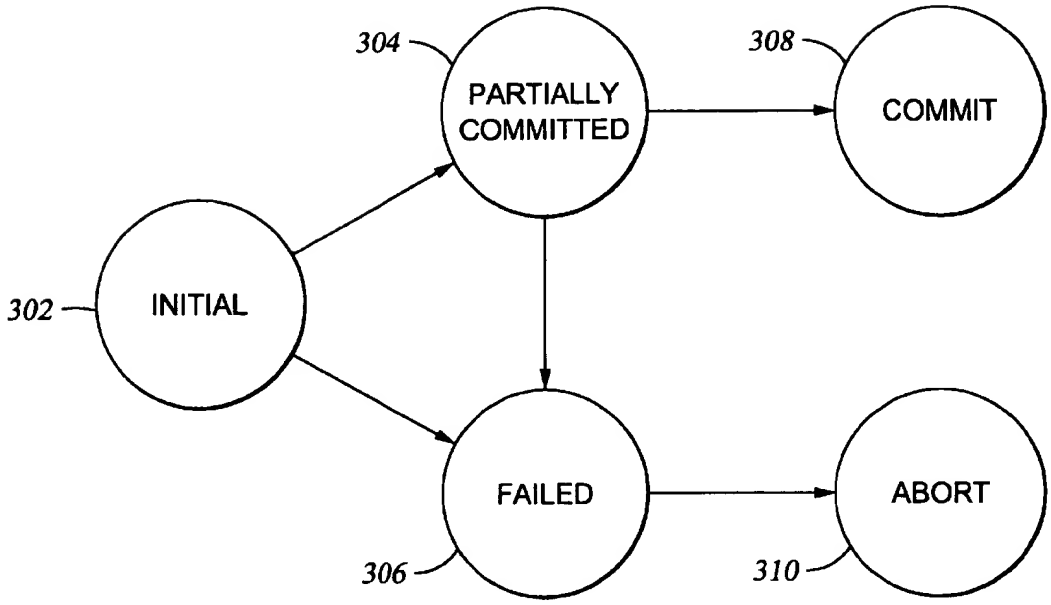

304
PARTIALLY COMMITTED
308
COMMIT
302
INITIAL
FAILED
ABORT
306
310

Fig. 3

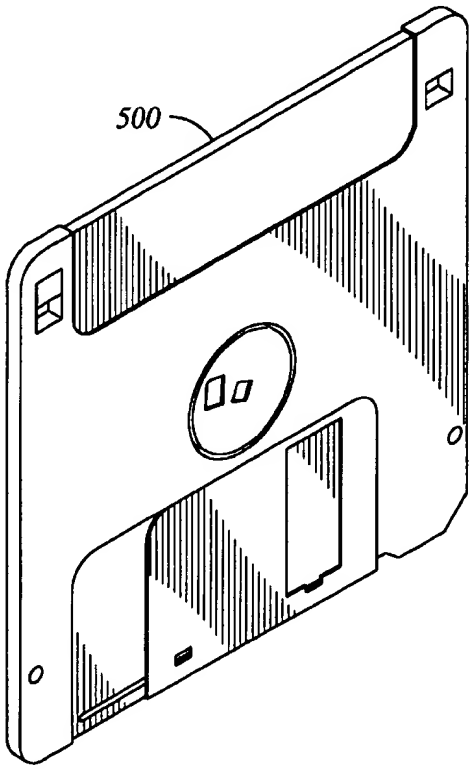

500

Fig. 5

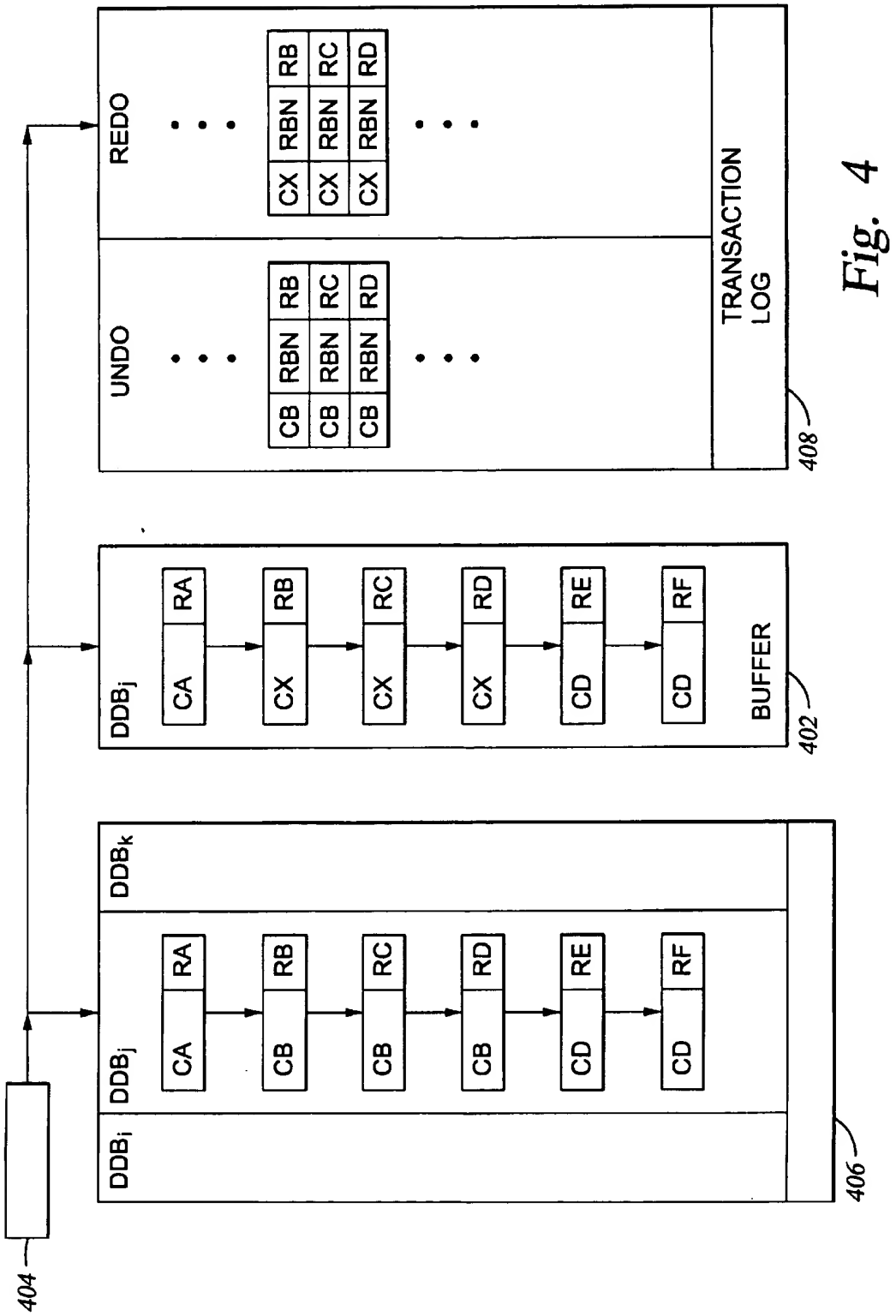

REDO
CX
RBN
RB
RC
RD
UNDO
CB
TRANSACTION LOG
408
DDBj
CA
RA
CX
RE
CD
RF
BUFFER
402
DDBk
DDBj
DDBi
406
404

Fig. 4

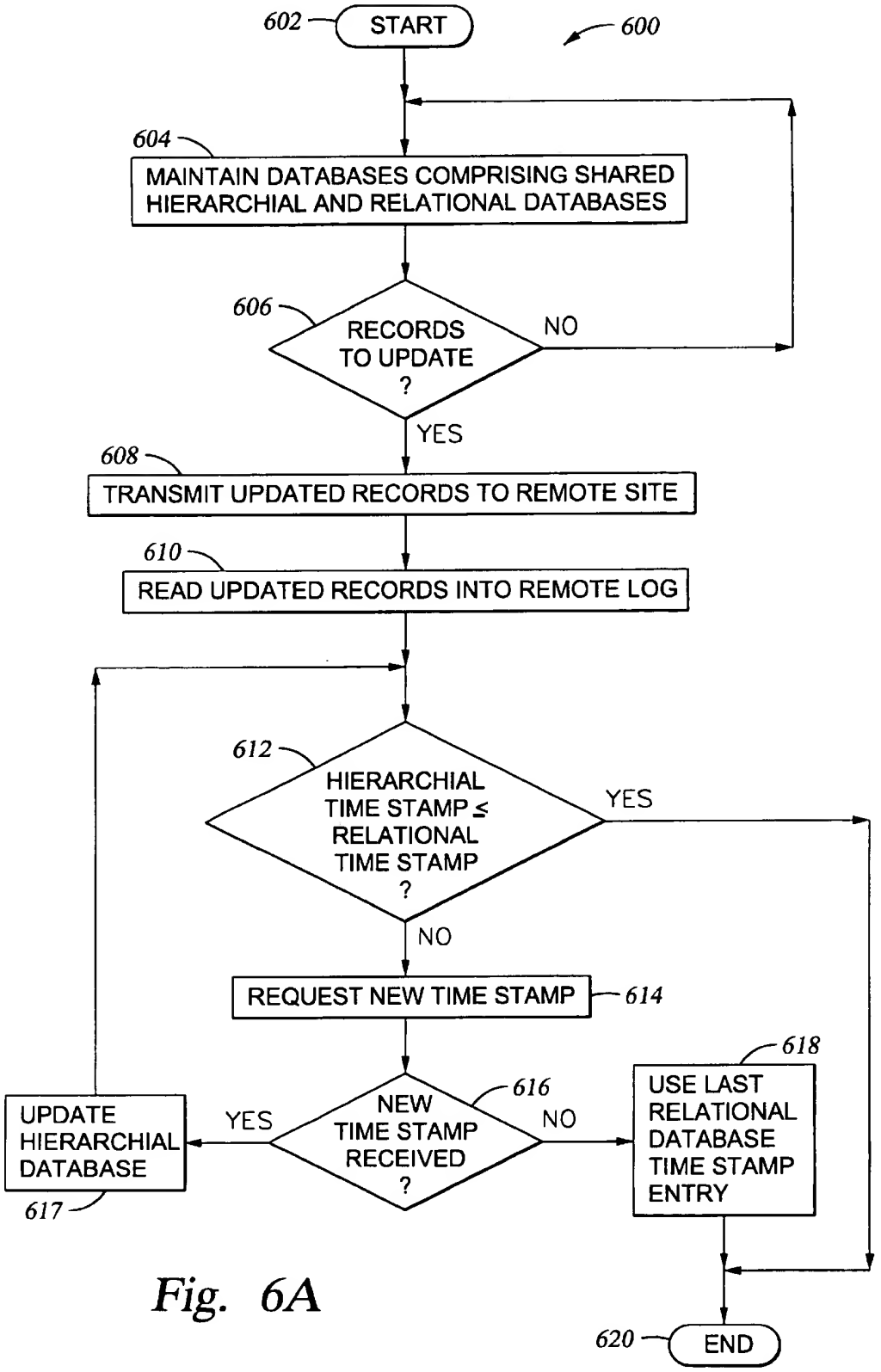

602
START
600
604
MAINTAIN DATABASES COMPRISING SHARED HIERARCHIAL AND RELATIONAL DATABASES
606
RECORDS TO UPDATE ?
NO
YES
608
TRANSMIT UPDATED RECORDS TO REMOTE SITE
610
READ UPDATED RECORDS INTO REMOTE LOG
612
HIERARCHIAL TIME STAMP ≤ RELATIONAL TIME STAMP ?
YES
NO
REQUEST NEW TIME STAMP
614
616
NEW TIME STAMP RECEIVED ?
YES
NO
UPDATE HIERARCHIAL DATABASE
617
618
USE LAST RELATIONAL DATABASE TIME STAMP ENTRY
620
END

Fig. 6A

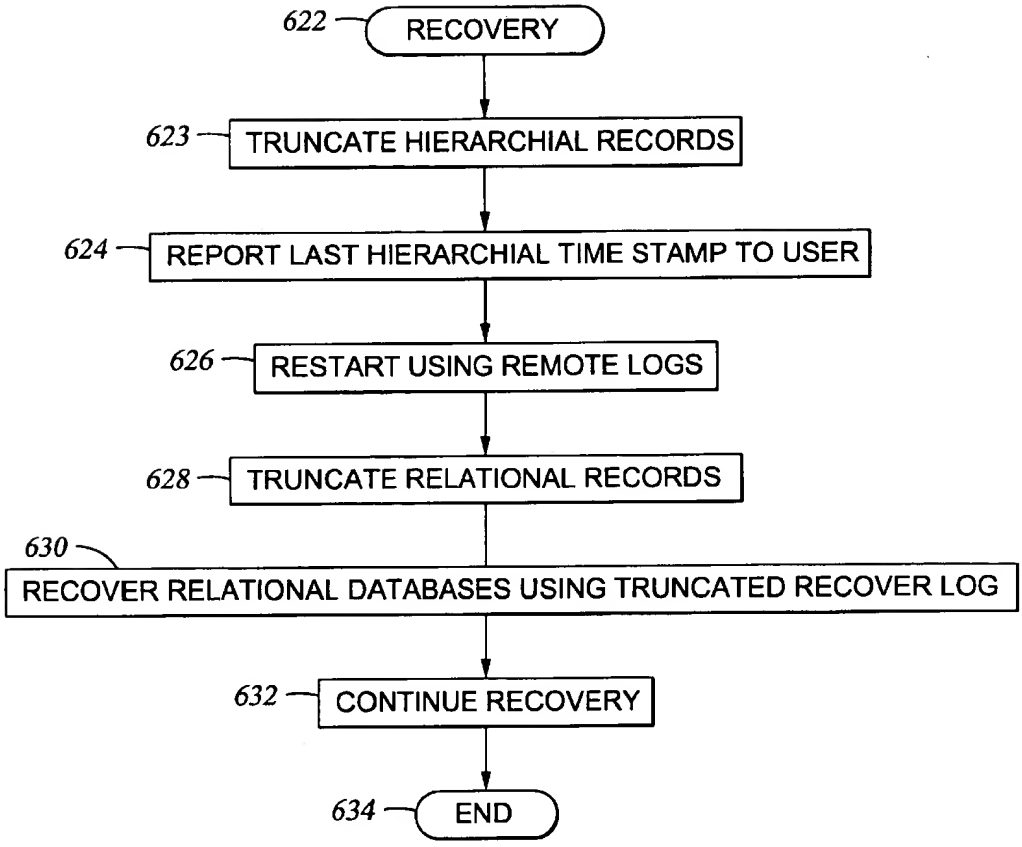

622
RECOVERY
623
TRUNCATE HIERARCHIAL RECORDS
624
REPORT LAST HIERARCHIAL TIME STAMP TO USER
626
RESTART USING REMOTE LOGS
628
TRUNCATE RELATIONAL RECORDS
630
RECOVER RELATIONAL DATABASES USING TRUNCATED RECOVER LOG
632
CONTINUE RECOVERY
634
END

Fig. 6B

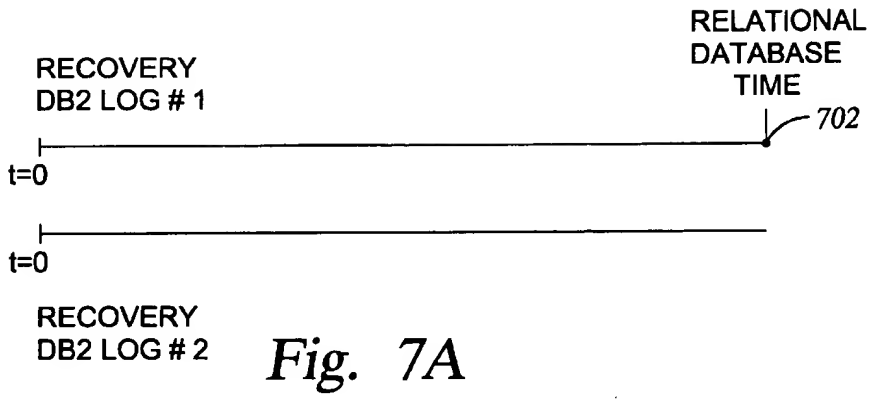


Fig. 7A

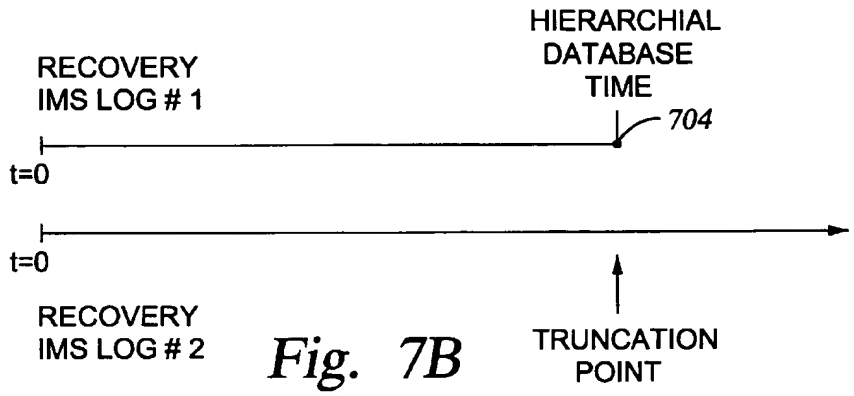


Fig. 7B

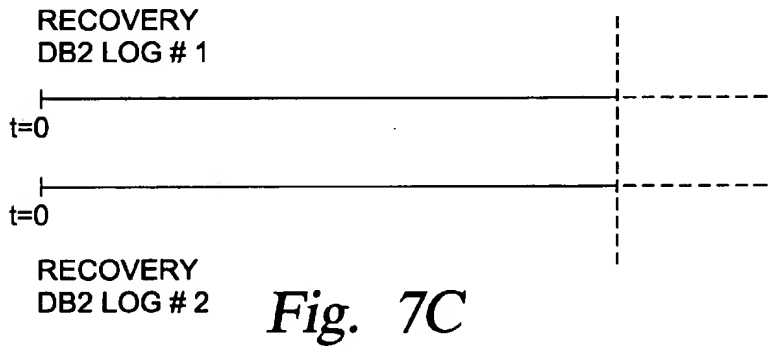


Fig. 7C

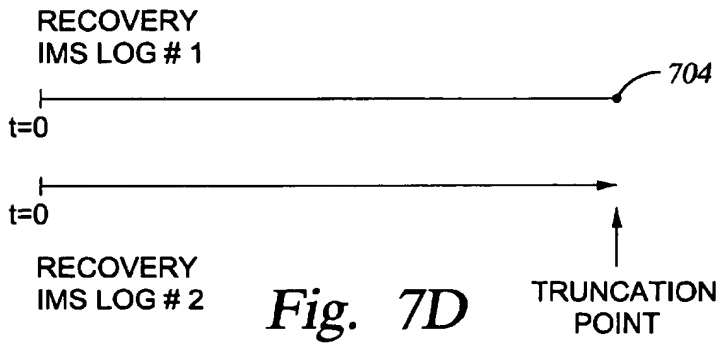


Fig. 7D

SYNCHRONIZING RECOVERY LOG HAVING TIME STAMP TO A REMOTE SITE FOR DISASTER RECOVERY OF A PRIMARY DATABASE HAVING RELATED HIERARCHIAL AND RELATIONAL DATABASES

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to disaster recovery in data processing systems using both hierarchial and relational databases. More particularly, the present invention relates to a method and apparatus to synchronize hierarchial and relational database recovery logs transmitted to a remote site for disaster recovery.

2. Description of the Related Art

Data processing systems typically require a large amount of data storage. Customer data, or data generated by users within the data processing system, usually occupy a great portion of this data storage. Effective data processing systems also provide back-up copies of this user data to insure against a loss of such data. For most businesses, any loss of data in their data processing systems is catastrophic, severely impacting the success of the business. To further protect customer data, some data processing systems extend the practice of making back-up recovery copies to provide disaster recovery. In disaster recovery systems, a recovery copy of the customer data is kept at a site remote from the primary storage location. If a disaster strikes the primary storage location, the customer data can be retrieved or "recovered" from the recovery copies located at the remote site.

A known method of providing disaster protection is to mirror, or shadow, the primary storage data at a remote storage site. Remote dual copy, or remote data duplexing, is one form of this data mirroring solution. In remote dual copy, additional storage devices are provided in the data processing system such that an additional copy of the primary data is written to a recovery storage device. Storage devices are coupled together to form duplex pairs, each duplex pair consisting of a primary and recovery storage device. The primary storage device is located at the primary storage location, while the recovery storage device is located at the remote site. When data is written to the primary storage device, the data processing system automatically copies the data to the recovery site.

Full volume copying is an alternate method for providing disaster recovery of a database. Full volume copying may use a storage management server to generate recovery storage volumes from the primary storage volumes. Commonly, a client-server configuration includes several clients connected to a single server. The clients create client files and transfer these files to the server. The server receives the client files and stores them on several attached storage devices. When used as a storage management system, the server manages the back-up, archival, and migration of these client files. By storing the client file on an attached storage device, the server creates a first back-up, or primary, copy of the client file. The server may, in turn, create additional back-up copies of the client file to improve the data availability and data recovery functions of the storage management system. Clients may vary from small personal computer systems to large data processing systems having a host processor connected to several data storage devices. The server can also range from a small personal computer to a large host processor.

To provide disaster recovery, the storage management server must generate a recovery copy of the client file and oversee the transmission of this recovery copy to a remote site. As a disaster recovery system, the server partitions the storage subsystem into a set of primary storage volumes and a set of remote, or off-site, recovery storage volumes. The off-site recovery volumes may contain removable media, so that they can be transported to the remote site. These volumes may be formatted using the same format or a different format from that used by the primary storage volumes for storing data and commands.

The server determines which client files need to be backed-up within the storage subsystem, how frequently these back-up copies should be made, or which set of the volumes should be transported to the remote site. The server or a separate controller may manage the off-site recovery storage volumes and determine which volumes are needed for disaster recovery. Off-site storage volumes no longer needed for disaster recovery may be reclaimed and reused. The server typically coordinates the reclamation and reuse of the recovery storage volumes. Successful reclamation and reuse of recovery volumes no longer needed for disaster recovery substantially improves the efficiency and performance of a disaster recovery system.

Incremental back-up techniques have evolved to improve the efficiency of disaster recovery systems. Using these techniques, only the user files new to the primary storage volume are copied to the recovery volumes since the last periodic back-up operation was completed. Thus, incremental back-up eliminates the unnecessary copying of files that remain unchanged since the previous back-up operation. As compared to full volume copying, incremental back-up reduces the number of partially filled storage volumes at the remote site. It also reduces the amount of duplicate files and duplicate volumes stored at the remote site, thereby simplifying the management of off-site recovery storage volumes.

Regardless of the recovery system used, these prior art recovery systems do not synchronize the remote recovery logs of the hierarchial and relational databases if independent transmission protocols are used. The update logs of the two database management systems—the hierarchial database and the relational database—are independently transmitted to the remote site. In essence, the logs for the hierarchial and relational database management systems operate as though each were a separate process even if the databases are related. If disaster strikes the primary site, the respective logs may not terminate at a consistent point.

For example, if a user has a system using an IMS and a DB2 system, current recovery systems do not synchronize the recovery logs. Although recovery logs may be kept, they are not synchronized with respect to time. As explained below, this may result in database inconsistencies if disaster strikes and the recovery logs at the remote site are used to recover the databases.

What is needed is a way to coordinate the disaster recovery techniques used by disparate database management systems. Remote log processing by the database management systems must be synchronized so that updates to related databases are consistent when a primary site disaster occurs. In addition, at least one of the database management systems needs to be able to update in real time the set of database copies it maintains at the remote site to keep them current with the primary set of databases to reduce the take-over time in the event of a disaster.

SUMMARY OF THE INVENTION

Broadly, the present invention concerns a method and apparatus to coordinate disaster recovery of related hierar-

chial and relational databases. Remote site recovery logs transmitted to a remote site are synchronized to expedite and insure consistent recovery in the event of a disaster at the primary site.

In one embodiment, the invention may be implemented to provide a method to synchronize logs at a remote site to recover logically different but related databases in the event a disaster occurs. The related databases in one embodiment are a hierarchial structured database and a relational structured database. A recovery log is used for recovery of database records, and a time stamp derived from a common time source may be assigned to each log record.

The update log records are transmitted to a recovery database management system where the log records are maintained in creation time sequence. If a disaster strikes the primary site, and recovery of data contained within the database is needed, the remote site recovery logs and databases may be used to recover some or all of the lost data. Based upon a comparison of the time stamp for the last received update record for the hierarchial database and the time stamp for the last received update record for the relational database, the logs are truncated to a common point in time. This truncation synchronizes the recovery logs and is used to generally ensure that the databases once recovered are consistent to a point as close as possible to the point at which the disaster occurred. For example, if an IMS and a DB2 database are being used, the IMS and DB2 recovery logs would be substantially synchronized by the truncation and, after recovery, their respective databases will be substantially consistent.

In another embodiment, the invention may be implemented to provide an apparatus for synchronizing the recovery logs. The apparatus comprises a storage management server coupled to a plurality of client systems. The server includes a storage manager, databases that may be shared, and a plurality of storage pools. The storage manager is coupled to the databases and the storage pools. A first storage pool, also referred to as a primary storage pool, contains a set of primary storage volumes for storing a primary copy of client files. The primary storage volumes may be located at a primary site. A secondary storage pool, also known as a recovery storage pool, contains a set of recovery storage volumes for storing a back-up copy of client files. The recovery volumes may comprise resident volumes located at the primary storage site and off-site volumes transported to a remote storage site.

In one embodiment, the storage manager receives client files from the client system and stores a primary copy of the client files in the primary storage pool. The storage manager also performs an incremental back-up operation by copying the newly created, or newly updated, client files in the primary storage pool to the recovery storage pool. The storage manager determines which recovery volumes are marked as off-site volumes and transported to the remote storage site. The storage manager also maintains a reference list, or index, within the database linking the primary copy of a client file to the recovery copy of the file within the copy storage pool. The storage manager automatically reclaims off-site recovery volumes no longer needed for disaster recovery before transporting the recovery volumes to the primary storage site.

In still another embodiment, the invention may be implemented to provide an article of manufacture comprising a data storage device tangibly embodying a program of machine-readable instructions executable by a digital data processing apparatus to perform method steps for synchronizing hierarchial and relational recovery logs transmitted to a remote site.

The invention affords its users with a number of distinct advantages. One advantage the invention provides is a way to synchronize the recovery logs of related databases where the databases use different database structures. Another advantage of the invention is that it ensures that the data on the related databases once recovered is consistent to the point at which a disaster occurs. The invention allows the complex problem of maintaining consistent databases to be accomplished in a manner which overcomes the time-consuming and error-prone methods currently practiced and putting data integrity at risk. For example, users of current methods generally write code to help in determining a consistent point to end the logs but the code depends on non-programming interfaces and thus can deteriorate with maintenance. In other words, as system software is replaced the code has to be updated or it may become dysfunctional.

This invention also provides a number of other advantages and benefits, which should be apparent from the following description of the invention.

BRIEF DESCRIPTION OF THE DRAWING

The nature, objects, and advantages of the invention will become more apparent to those skilled in the art after considering the following detailed description in connection with the accompanying drawings, in which like reference numerals designate like parts throughout, wherein:

FIG. **1** is a block diagram of a data processing system showing a plurality of client systems coupled to a storage management server;

FIG. **2** is a block diagram of the storage management server in FIG. **1** providing off-site storage volumes for disaster recovery;

FIG. **3** is a block diagram illustrating a procedure commonly followed in committing updated data to a database;

FIG. **4** is a block diagram illustrating details of the database management system of FIG. **2**;

FIG. **5** is a perspective view of an exemplary signal-bearing medium in accordance with the invention;

FIGS. **6A** and **6B** are a flowchart of an operational sequence for synchronizing the hierarchial and relational logs transmitted to a remote site for recovering related databases; and

FIGS. **7A** through **7D** illustrate the truncation of respective log records as discussed with respect to task **628** shown in FIG. **6B**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Hardward Components & Interconnections

One aspect of the invention concerns an apparatus **100** for synchronizing recovery logs transmitted to a remote site for disaster recovery of related databases which may be embodied by various hardware components and interconnections as described in FIG. **1**. Throughout this discussion, "recovery" refers to disaster recovery of a primary site with a remote site.

The apparatus **100** employs multiple client systems **102** coupled to a server system (primary site) **103**. The primary site **103** includes a storage manager **104** coupled to a server database **106**. The database may comprise multiple databases using similar or dissimilar formatting. The storage manager **104** is further coupled to a plurality of primary storage pools **110** and a cache **130**. A storage pool **110** consists of a plurality of storage devices, either DASD,

optical disk, or magnetic tape devices. All storage devices within the primary storage pool **110** may be, but are not necessarily, identical in type and format. The server database is further coupled to a set of recovery volumes **220** shown in FIG. **2** and providing a back-up for the server database **106**. In one embodiment, the recovery volumes **220** are located at a site remote from the primary storage site.

Each client system **102** creates original user data files, or client files, which are stored within the corresponding client system if the client system is provided with a storage system. Regardless, the client systems **102** transfer client files to the server system **103**. Transferring client files to the primary site **103** inherently provides a back-up mechanism within the server for original client files stored within the client system. The storage manager **104** directs the client file to a storage device, or storage volume, within a primary storage pool **110**. The primary storage pool stores a primary copy of the client files. The storage manager **104** maintains a log **105** within the server database **106** listing the files stored within the primary storage pool **110** and the cache **130** of the primary site **103**. Once the client file is stored within a primary storage pool **110**, the storage manager **104** updates the server database **106** and logs the updates in log **105** at the primary site **103**.

In one embodiment, the server system **103** might also include the cache **130**. This cache is used by the storage manager **104** to coordinate temporary storage of data copied from nonvolatile storage during a restart recovery process. In one embodiment, the cache **130** is volatile storage. In another embodiment, the cache is nonvolatile but erasable. Regardless, if the cache **130** becomes full during a restart recovery process, over-flow data may be written to another cache, for example, cache (not shown) located in the storage manager **104**.

In one embodiment, the server system might also generate an additional back-up recovery copy of the client file and store this recovery copy on a storage device, or storage volume, within a remote site **200** shown in FIG. **2**. The storage manager **104** coordinates this operation. Once the additional recovery copy is created, the storage manager **104** updates the log **105** of server database **106** to catalog the recovery copy of the client file. In addition, the log entry within the server database corresponding to the recovery copy includes cross-reference to the primary copy of the client file. Thus, the primary copy in the primary storage pool **110** is linked to the recovery copy.

FIG. **2** shows a server (referred to as remote or off-site) **200** of FIG. **1** configured as a remote disaster recovery system. The disaster recovery system includes a remote storage pool **220** within the remote site **200**. The remote site server **200** also includes the controller **202** and a recovery log **204**. In addition, the remote storage pool **220** may contain copies of the data volumes contained in the primary storage volumes located within the primary storage pools **110** and any other resident copy storage volumes within the primary storage site **103**. The remote site **200** contains remote storage pool **220** which are copy storage volumes that have been transferred from the primary storage site **103** at the direction of the storage manager **104**.

The storage manager **104** determines which primary storage pool volumes **111**–**120** should be used for back-up storage and recovery, and then copies and marks the designated volume in the server database **103** as an off-site storage volume. The copies are then delivered to the remote site **200**. The storage manager **104** uses the server database **106** to monitor the recovery copies of the client files stored on the off-site storage volumes **221**–**229**. The storage manager **104** can then determine when an off-site volume is no longer needed for disaster recovery, and marks this volume for deletion or transfer back to the primary storage site **103**. Further, as each database transaction is executed and each database recovery copy is made, an entry is made by the storage manager **104** in the log **105** to keep track of each transaction and data update. In another embodiment, back-up volumes to the primary storage pool **110** may be kept at the primary site **103** in addition to the remote storage pool **220** back-up copies housed at the remote site **200**.

Regardless of whether primary storage pool volume **111** through **120** is marked and copied as an off-site volume and then physically transported to the remote site, or whether the contents of the copy of the primary storage pool volumes is transmitted to the remote site via a data link, the remote site **200** shown in FIG. **2** processes the data in a similar manner. When updated data is received by the controller **202** of the remote system **200**, the controller decides which of the off-site storage volumes **220** and respective recovery logs will receive the updated data. If an existing file on a remote site volume **221** through **229** is to be updated, the controller **202** routes the updated data to the respective volume and log. If no current file exists that needs to be updated, the controller **202** may allocate the file to any of the storage volumes **220**. As the updated data is received, the controller **202** also updates a recovery log **204**.

The log **204** acts as index for the data records being maintained on the remote storage pool **220** volumes. The recovery log **204** may also perform other functions as discussed below. The transfer of data from the primary storage site **103** to the remote site **200** is coordinated by the storage manager **104** and the controller **202**.

Despite the specific foregoing description, ordinarily skilled artisans (having the benefit of this disclosure) will recognize that the apparatus discussed above may be implemented in a machine of different construction, without departing from the scope of the invention.

Operation

In addition to the various hardware embodiments described above, a different aspect of the invention concerns a method for recovering related databases using the present invention.

Transaction Execution

The apparatus shown in FIGS. **1** and **2** are used in one embodiment to maintain related databases. For example, the server database **106** may contain multiple databases where the databases do not share a similar logical structure or format. Regardless, each of the databases is intended to contain data which reflects a real state of the information that database is supposed to capture. When the databases reflect what is the current state of the data, the databases are deemed consistent.

To better understand the recovery system of the present invention, the following terminology should be helpful. A database transaction is a program unit whose execution preserves the consistency of the database. For example, if before a transaction executes the database is in a consistent state then, when the next transaction completes its execution, the database remains in a consistent state. The transaction assesses and possibly updates various data items contained in the database. Each of the data items may be read by the transaction and is written by the transaction if it updates that data item. A transaction is the work that occurs between the beginning of a unit of work and a commit or abort as defined below.

As reflected in FIG. 3, a transaction block diagram is shown. The transaction is shown in its initial state in task **302**. When it reaches its last statement, it enters a partially committed state in task **304**. At this point, the transaction has completed its execution, but it is still possible that it may have to be aborted, which includes rolling back the non-committed data, since the actual updates may not have been output to storage yet, and thus a hardware failure may preclude its successful completion. In one embodiment of the present invention, writing to storage takes place only after a transaction has entered the commit stage as shown in task **308**. As discussed below, one way to implement such a scheme is to store any value associated with such writing to storage temporarily in a nonvolatile storage, and to perform the actual writes only at commit time **308**. A committed transaction will then be able to complete its write except in the case of hardware storage failures.

The transaction enters the failed state in task **306** if it is determined that the transaction cannot proceed with normal execution, that is, for example, due to hardware or logical errors. If failure occurs, the transaction must be rolled-back. Once a rollback has occurred, the transaction enters the aborted state in task **310**.

As applied to this invention and as illustrated in FIG. **4**, "forward" database updating is performed by using REDO records of a transaction log **408**. As an example, assume that records, identified by record numbers RA–RF are included in the defined data block j (DDBj) which is included in a data set stored in a database **406**. Assume next a transaction against records RB–RD of DDBj, and that the records are to be updated as a result of the transaction. For updating, storage manager **404** transfers the page DDBj to a buffer **402** where the records RB–RD are updated, in each case by changing record data CB to CX. During the processing of these records, log entries are made in the sequence in which the records were updated. Each record update results in the writing of an UNDO and REDO record to the transaction log **408**. For example, the transaction log entry for updating record RB includes an UNDO record with a field containing the record data before updating (CB), a field identifying the record (RB), and a field containing a relative block number (RBN) which identifies the location of the primary storage page containing the record. The REDO component of the log record includes a field containing the update of the record data (CX) and the RBN and RB fields identifying, respectively, the page and the record. Assuming completion of the transaction or set of transactions which update records RB–RD, the storage manager **404** will write the buffer page to the primary storage location for DDBj. Later, if recovery is required, the recovery process will use the transaction log REDO records to update records RB–RD.

Forward and "backward" updating recovery functions are available in database products manufactured by the IBM Corporation. Forward and backward updating are supported in the database manager of the IBM DB2 and IMS/ESA program products. In these products, UNDO records are logged in addition to REDO records. If a transaction associated with a REDO record ABORTs, the record data from the UNDO record is logged as a REDO record and is applied to the database, thus "backing-out" the original update to the record. In the FAST PATH database manager of the IBM IMS/ESA program product used in one embodiment, UNDO records are not logged. Rather, changes are withheld from the database until REDO records have been logged and the updating transactions have completed. Successful completion of the transaction is indicated by a COMMIT operation which is recorded by a COMMIT record in the transaction log. If the transaction is abnormally terminated, an ABORT record is entered in the log.

After the server database **106** and the storage pools of server **103** have been updated by a transaction, the off-site recovery storage volumes of remote site **200** may be updated, as described below with respect to the method of the invention. However, because of the related databases in one embodiment of the present invention where each database may not employ a common logical structure or format, the present invention provides a method to coordinate recovery of the related databases and the logs used for recovery. The method of the present invention is described in detail as to its overall sequence of operation below.

Overall Sequence of Operation

FIGS. **6A** and **6B** show a sequence of method steps **600** to illustrate one example of the method aspect of the present invention. For ease of explanation, but without any limitation intended thereby, the example of FIGS. **6A** and **6B** are described in the context of the apparatus **100** described above. The steps are initiated in task **602**, when a primary database system linked to a remote recovery facility processes transactions. The primary database comprises a hierarchial database and a relational database. In the one embodiment, the hierarchial database is a IMS database and the relational database is a DB2 database.

The related hierarchial and relational databases are maintained in task **604**. Maintaining the primary database comprises updating the database with changes to the data stored therein and continually updating that data. Several techniques are known in the art for maintaining the primary database in a consistent state. The primary database may incorporate separate logs for the hierarchial database and the relational database, and may implement a plurality of logs for each. The update log information may be transmitted to a remote site to insure recovery if disaster recovery is required by the primary database.

In maintaining a database, an entry is made to each update log and a time stamp is assigned to each entry. Commonly, the time stamp is assigned by the storage manager **104** shown in FIG. **1** or the controller **202** shown in FIG. **2**, however, those skilled in the art will realize that any processor used in conjunction with the primary database may assign the time stamp. The time stamp represents the creation time sequence for a given entry, that is, the time at which the entry was created in a respective log.

When entries to the update log are complete in task **606**, the update log records are transmitted to the remote site for backup protection. If no records need to be updated as the result of a transaction, the method returns to task **604** and continues maintaining the database.

If hierarchial log records from task **608** are transmitted from the primary database to the recovery database, they are processed only up to the point that data has been captured on the relational database log volumes at the remote site. The point is established by comparing entry time stamps. This assures that in the event of a primary site disaster the truncation point for the hierarchial tracking logs will be earlier in time than the end point of the relational tracking logs. Using the time stamp of task **604**, the updates to the records are transmitted in task **608** to the remote site. In one embodiment, the record updates may be transmitted to the recovery database's management system including controller **202** in parallel in task **610**.

As part of maintaining the recovery resources, the storage manager **104** or the controller **202** keeps track of the time stamp of the last hierarchical and relational log entries. The hierarchical time stamp is compared to the last relational log

in task **612**, then the updates are applied to the hierarchical databases. If the hierarchical time stamp is not less than or equal to the relational log time stamp, then the method **600** requests a new time stamp for the last entry to the relational log in task **614**. The request seeks to determine the time stamp of the last entry to the relational database logs.

If a new time stamp for the last entry to the relational database (relational time stamp) is received within a designated period of time in task **616**, then the method **600** again compares the hierarchial database time stamp in task **617** (hierarchial time stamp) with the relational time stamp in task **612**, and if the hierarchial time stamp is less than or equal to the relational time stamp, the updated log records are applied to the remote site **200** hierarchial database. By assuring that the hierarchial time stamp is less than or equal to the relational time stamp, the update logs for the hierarchial database and the relational database are able to be synchronized for later processing (truncation).

Returning to task **616**, if a new hierarchial time stamp has not been received within a predetermined time period, then the method **600** uses the last relational database time stamp in task **618** as the point in time to stop applying updates to the remote site hierarchial databases. Transaction updates to the databases which occurred after this time stamp are not applied unless a new relational time stamp is received in task **618**. This routine maintenance of the databases and related logs ends in task **620**. If disaster recovery is required, this time stamp represents the end time stamp of the hierarchial relational logs. The hierarchial logs are truncated at the point represented by this time stamp in task **623** shown in FIG. **6B**.

If disaster strikes and a takeover of the primary site functions by the remote site is required, the remote recovery method of the invention is implemented as shown in FIG. **6B** beginning at task **622**. For example, the primary site may be rendered unusable due to fire, flood or earthquake.

If disaster recovery is required, the last hierarchial time stamp recorded on the recovery log is reported in task **624**. This time stamp is used to coordinate the remote site **200** recovery. After the time stamp is reported, the remote site's controller **202** may automatically initiate recovery. In another embodiment, the time stamp is reported to a user, such as a database administrator, and the user initiates the recovery. For example, the last hierarchial time stamp may be displayed on a device such as a monitor, printed-out on a printer, audibly output, or otherwise communicated to the user. The communication device used may be interfaced with the client **102** shown in FIG. **1** and/or the remote site **200** shown in FIG. **2**. In either case, the hierarchial time stamp is reported as the "recovery timestamp" in task **624** and used to synchronize the recovery logs to that point in time for recovery and restart purposes in task **626**.

In one embodiment, the end user may only use the last hierarchial time stamp as the recovery time stamp. The time stamp selected is used as a truncation point to synchronize the recovery log records at the selected point in time in task **628**. These recovery log records, comprising the hierarchial and relational database recovery logs, are used to restart and recover the primary database at the remote site in task **630**.

Truncation of the recovery log records at the remote site is best understood referring to FIGS. **7A** through **7D**. FIGS. **7A** and **7B** show a time line for record entries for the preferred embodiment DB2 logs and IMS logs received at the remote site, respectively, used in conjunction with the DB2 and IMS databases. At time 0, no record entries have been made. As time increases, indicated by the line increasing to the right, an increasing number of record entries have been made to each log. FIG. **7A** shows the DB2 logs having an end time marked at the point-in-time **702**. FIG. **7B** shows the IMS LOG **#1** having an end time marked at a point-in-time **704** and IMS LOG **#2** continuing past time **702**.

The recovery logs are constructed such that the hierarchial database recovery time lags the relational database recovery time, that is, that processing of entries in the IMS logs shown in FIG. **7B** always lag the recovery time of the DB2 logs shown in FIG. **7A**. Comparison of FIGS. **7A** and **7B** indicate that the hierarchial database recovery time **704** for the IMS logs does lag the relational database recovery time **702**. As shown in FIGS. **7C** and **7D**, respectively, the DB2 logs are truncated in the present invention to the hierarchial database recovery time represented by time stamp **704**, and the hierarchial logs are also truncated to that point in time. The DB2 log data contained on the DB2 logs at a time after recovery time stamp **704** is not used in the data recovery method **600**. This "truncation" results in the DB2 log recovery time equaling the IMS log recovery time which synchronizes the logs used to recover the hierarchial and relational databases.

After truncation of the logs, recovery continues in task **632**. During recovery, the primary site is not available and recovery is coordinated from the remote site **200**. Recovery continues when the hierarchial and relational databases are restarted at the remote site **200**. The method ends in task **634**.

During remote site recovery, update log records are written to the logs at the remote site. The relational databases are not maintained in real time as update log records are received. They are recovered after a disaster has occurred using normal recovery such as generally discussed in U.S. Pat. No. 5,615,329, issued Mar. 25, 1997, entitled "REMOTE DATA DUPLEXING," incorporated herein by reference and assigned to the assignee of the present invention, or using other commonly practiced recovery utilities. Once the remote site **200** databases have been recovered, the recovered databases become the "primary" database because the original primary database is no longer useable. These remote logs are used to restore the primary database to its prior state as of the time reported in task **624** used for recovery purposes.

In one embodiment, shadowing of the relational remote site databases may occur by running the recovery utility periodically at the remote site **200**. The hierarchial remote site databases are shadowed in real time but the relational remote site databases are not because the remote relational logs are subject to truncation—allowing the hierarchial recovery log to always lag the relational recovery log—as discussed above. This periodical "shadowing" of the relational remote site **200** databases reduces the time required to recover the relational databases if a primary site disaster strikes.

Signal-Bearings Media

Such a method may be implemented, for example, by operating the apparatus **100** to execute a sequence of machine-readable instructions. These instructions may reside in various types of signal-bearing media. In this respect, one aspect of the present invention concerns a programmed product, comprising signal-bearing media tangibly embodying a program of machine-readable instructions executable by a digital data processor to perform a method to synchronize recovery logs transmitted to a remote site to recover related databases having dissimilar logical structures or formats.

This signal-bearing media may comprise, for example, RAM (not shown) contained within the apparatus **100**. Alternatively, the instructions may be contained in another signal-bearing media, such as a magnetic data storage dis-

kette **500** as shown in FIG. **5**, directly or indirectly accessible by the apparatus **100**. Whether contained in the apparatus **100** or elsewhere, the instructions may be stored on a variety of machine-readable data storage media, such as DASD storage (e.g., a conventional "hard drive" or a RAID array), magnetic tape, electronic read-only memory (e.g., ROM, CD-ROM, EPROM, or EEPROM), an optical storage device (e.g., CD-ROM, WORM, DVD, digital optical tape), paper "punch" cards, or other suitable signal-bearing media including transmission media such as digital and analog and communication links and wireless. In an illustrative embodiment of the invention, the machine-readable instructions may comprise lines of compiled C, C++, or similar language code commonly used by those skilled in the programming for this type of application arts.

Other Embodiments

While there have been shown what are presently considered to be preferred embodiments of the invention, it will be apparent to those skilled in the art that various changes and modifications can be made herein without departing from the scope of the invention as defined by the appended claims.

What is claimed is:

1. A method for synchronizing a recovery log to a remote site for disaster recovery of a primary database having related hierarchial and relational databases, comprising:

processing hierarchial records and relational records of the recovery logs wherein a time stamp of each hierarchial record is ≦ to a time stamp for a last entered relational record, and wherein a time stamp is used to mark a point in time and a hierarchial record reflects hierarchial database data and a relational record reflects relational database data.

2. The method recited in claim **1**, further comprising transmitting the hierarchial records to the remote site independent from the transmission of the relational database.

3. The method recited in claim **2**, further comprising maintaining a hierarchial database in real time at the remote site that shadows the primary site hierarchial database.

4. The method recited in claim **3**, further comprising:

(a) continually comparing, in sequence by time stamp, the time stamp for each hierarchial record to a last entered time stamp for a relational record; and

(b) processing recovery log records:

(1) if the time stamp for the hierarchial record is ≦ to the last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record for comparison;

(2) if the time stamp for the hierarchial record is not ≦ to the last entered time stamp for the relational record, requesting a new last entered time stamp for a relational record;

(i) if the time stamp for the hierarchial record is ≦ the new last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record;

(ii) if no new time stamp is received within a specified period of time, periodically requesting a new last entered time stamp for a relational record until a new last entered time stamp for a relational record is obtained; and

(iii) suspending hierarchial record processing until a new last entered time stamp for a relational record is obtained.

5. The method recited in claim **4**, further comprising truncating a recovery log when a primary site disaster occurs at a point in time substantially coinciding with the last time stamp used for processing recovery log records.

6. The method recited in claim **5**, further comprising:

truncating first the hierarchial records at the point in time;

reporting a truncation time; and then

truncating the relational records at the truncation time.

7. The method recited in claim **6**, further comprising processing the hierarchial and the relational records entered into the recovery logs and having an earlier time stamp than the truncation time.

8. The method recited in claim **7**, further comprising maintaining a primary site, the primary site including a database and logs, the logs comprising updated records to be transferred to the database, the database including related hierarchial and relational databases, the remote site including recovery logs and a recovery database, the recovery database including related hierarchial and relational databases, the recovery logs comprising updated records to be transferred to the recovery database, the recovery site used to take over the database functions of the primary site if a disaster occurs at the primary site.

9. A signal-bearing medium tangibly embodying a program of machine-readable instructions executable by a digital processing apparatus to perform a method for synchronizing update recovery logs transmitted to a remote site for disaster recovery of a primary database having related hierarchial and relational databases, said method comprising:

processing hierarchial records and relational records of the recovery logs wherein a time stamp of each hierarchial record is ≦ to a time stamp for a last entered relational record, and wherein a time stamp is used to mark a point in time and a hierarchial record reflects hierarchial database data and a relational record reflects relational database data.

10. The medium recited in claim **9**, the method further comprising transmitting the hierarchial records to the remote site independent from the transmission of the relational records.

11. The medium recited in claim **10**, the method further comprising maintaining a hierarchial database in real time at the remote site that shadows the primary site hierarchial database.

12. The medium recited in claim **11**, the method further comprising:

(a) continually comparing, in sequence by time stamp, the time stamp for each hierarchial record to a last entered time stamp for a relational record; and

(b) processing recovery log records:

(1) if the time stamp for the hierarchial record is ≦ to the last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record for comparison;

(2) if the time stamp for the hierarchial record is not ≦ to the last entered time stamp for the relational record, requesting a new last entered time stamp for a relational record;

(i) if the time stamp for the hierarchial record is ≦ the new last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record;

(ii) if no new time stamp is received within a specified period of time, periodically requesting a new last entered time stamp for a relational record until a new last entered time stamp for a relational record is obtained; and

(iii) suspending hierarchial record processing until a new last entered time stamp for a relational record is obtained.

13. The medium recited in claim 12, the method further comprising truncating a recovery log when a primary site disaster occurs at a point in time substantially coinciding with the last time stamp used for processing recovery log records.

14. The medium recited in claim 13, the method further comprising:

truncating first the hierarchial records at the point in time;

reporting a truncation time; and then

truncating the relational records at the truncation time.

15. The medium recited in claim 14, the method further comprising processing the hierarchial and the relational records entered into the recovery logs and having an earlier time stamp than the truncation time.

16. The medium recited in claim 15, the method further comprising maintaining a primary site, the primary site including a database and logs, the logs comprising updated records to be transferred to the database, the database including related hierarchial and relational databases, the remote site including recovery logs and a recovery database, the recovery database including related hierarchial and relational databases, the recovery logs comprising updated records to be transferred to the recovery database, the recovery site used to take over the database functions of the primary site if a disaster occurs at the primary site.

17. An apparatus used to synchronize recovery logs transmitted to a remote site for disaster recovery of a primary database, the apparatus comprising:

a primary site including primary data storage, the primary storage used for storing the primary database, the primary storage including at least one log comprising updated records for transfer to the primary database, the primary database including a hierarchial database and a relational database;

a remote site including remote site data storage, the remote site storage used for maintaining the remote database and including at least one recovery log;

a link communicatively coupling the primary site with the remote site;

a processor communicatively linked to the primary site and the remote site, the processor executing a series of commands to synchronize recovery logs transmitted from the primary site to the remote site by:

processing hierarchial records and relational records of the recovery logs wherein a time stamp of each hierarchial record is ≦ to a time stamp for a last entered relational record, and wherein a time stamp is used to mark a point in time and a hierarchial record reflects hierarchial database data and a relational record reflects relational database data.

18. The apparatus recited in claim 17, the processor further executing commands for transmitting the hierarchial records to the remote site independent from the transmission of the relational records.

19. The apparatus recited in claim 18, the processor further executing commands for maintaining a hierarchial database in real time at the remote site that shadows the primary site hierarchial database.

20. The apparatus recited in claim 19, the processor further executing commands for:

(a) continually comparing, in sequence by time stamp, the time stamp for each hierarchial record to a last entered time stamp for a relational record; and

(b) processing recovery log records:

(1) if the time stamp for the hierarchial record is ≦ to the last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record for comparison;

(2) if the time stamp for the hierarchial record is not ≦ to the last entered time stamp for the relational record, requesting a new last entered time stamp for a relational record;

(i) if the time stamp for the hierarchial record is ≦ the new last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record;

(ii) if no new time stamp is received within a specified period of time, periodically requesting a new last entered time stamp for a relational record until a new last entered time stamp for a relational record is obtained; and

(iii) suspending hierarchial record processing until a new last entered time stamp for a relational record is obtained.

21. The apparatus recited in claim 20, the processor further executing commands for truncating a recovery log when a primary site disaster occurs at a point in time substantially coinciding with the last time stamp used for processing recovery log records.

22. The apparatus recited in claim 21, the processor further executing commands for:

truncating first the hierarchial records at the point in time;

reporting a truncation time; and then

truncating the relational records at the truncation time.

23. The apparatus recited in claim 22, the processor further executing commands for processing the hierarchial and the relational records entered into the recovery logs and having an earlier time stamp than the truncation time.

24. An apparatus for synchronizing update recovery logs at a remote site for recovery of related hierarchial and relational databases, the apparatus comprising:

a means for processing hierarchial records and relational records of the recovery logs wherein a time stamp of each hierarchial record is ≦ to a time stamp for a last entered relational record, and wherein a time stamp is used to mark a point in time and a hierarchial record reflects hierarchial database data and a relational record reflects relational database data;

means for maintaining a hierarchial database in real time at the remote site that shadows the primary site hierarchial database; and

means for transmitting the hierarchial records to the remote site independent from the transmission of the relational records.

25. The apparatus recited in claim 24, including:

(a) continually comparing, in sequence by time stamp, the time stamp for each hierarchial record to a last entered time stamp for a relational record; and

(b) processing recovery log records:

(1) if the time stamp for the hierarchial record is ≦ to the last entered time stamp for the relational record, processing the hierarchial record and selecting a next hierarchial record for comparison;

(2) if the time stamp for the hierarchial record is not ≦ to the last entered time stamp for the relational record, requesting a new last entered time stamp for a relational record;

(i) if the time stamp for the hierarchial record is ≦ the new last entered time stamp for the relational

record, processing the hierarchial record and selecting a next hierarchial record;

(ii) if no new time stamp is received within a specified period of time, periodically requesting a new last entered time stamp for a relational record until a new last entered time stamp for a relational record is obtained; and

(iii) suspending hierarchial record processing until a new last entered time stamp for a relational record is obtained; and

truncating a recovery log when a primary site disaster occurs at a point in time substantially coinciding with the last time stamp used for processing recovery log records.

26. The apparatus recited in claim **25** wherein the truncation means truncating first the hierarchial records at the point in time, reporting a truncation time, and then truncating the relational records at the truncation time.

27. The apparatus recited in claim **26**, including a means for processing the hierarchial and the relational records entered into the recovery logs and having an earlier time stamp than the truncation time.

* * * * *